Knobbe Martens Olson & Bear LLP

Intellectual Property Law

2040 Main Street
Fourteenth Floor
Irvine, CA 92614
Tel 949-760-0404
Fax 949-760-9502
kmob.com

ORIGINAL WILL FOLLOW VIA:

- ☐ MAIL
- ☐ INTERNATIONAL AIRMAIL
- ☐ COURIER
- ■ WILL NOT FOLLOW
- ☐ HAND DELIVERY
- ☐ WITH ENCLOSURES
- ☐ WITHOUT ENCLOSURES

Facsimile Transmittal Sheet

Confidentiality Notice:

The documents accompanying this facsimile transmission contain confidential information which may be legally privileged. The information is intended only for the use of the recipient named below. If you have received this facsimile in error, please immediately notify us by telephone to arrange for return of the original documents to us. Any disclosure, copying, distribution or the taking of any action in reliance on the contents of this faxed information is strictly prohibited.

TO: Examiner Kuen S. Lu, Group Art Unit 2177

FIRM: UNITED STATES PATENT AND TRADEMARK OFFICE

FACSIMILE NO: 703-872-9306

YOUR REFERENCE: U.S. Patent Application No. 10/050,675 OUR REFERENCE: FNDSTN.013A

FROM: Ted M. Cannon, Reg. No. 55,036

TOTAL NUMBER OF PAGES: **208** (INCLUDING COVER SHEET).

OPERATOR: Kristin Eldred DATE: June 30, 2004 TIME:

IF YOU DID NOT RECEIVE ALL OF THE PAGES, PLEASE CALL 949-721-2849 IMMEDIATELY.

MESSAGE: Dear Examiner Lu:

In accordance with our telephone conference. I am attaching copies of the documents you requested. Please call Jerry T. Sewell at 949-721-2849 or Kristin Eldred at 949-721-2846 if you need additional information.

JTS-20289.DOC:kc
20040630

Steven Brower, State Bar #93568
Brian Barrow, State Bar #177906
STEPHAN, ORINGHER RICHMAN & THEODORA
A Professional Corporation
535 Anton Boulevard, Suite 800
Costa Mesa, California 92626
Telephone: (714) 241-0420

Attorneys for Defendants

SUPERIOR COURT OF THE STATE OF CALIFORNIA

FOR THE COUNTY OF ORANGE

FOUNDSTONE, INC., Plaintiff, vs. NT OBJECTIVES, INC., et al., Defendants.	CASE NO: 02CC15350 Judge David H. Brickner Dept. C17 **DECLARATION OF DAN KUYKENDALL IN OPPOSITION TO OSC RE: PRELIMINARY INJUNCTION** Date: October 25, 2002 Time: 3:00 p.m. Dept: C17

I, Dan Kuykendall, declare:

1. The facts stated herein are of my own personal knowledge except for those set forth on information and belief, and as to any such facts, I have a basis for believing that they are true. If called upon to testify as a witness I could and would competently do so as set forth herein.

ROLE IN THIS LITIGATION

2. I am individually named as a defendant in this matter.

3. I was employed by Plaintiff Foundstone, Inc. from August 2001 until July 1, 2002, as more fully set forth below. I was never an officer or director of that company.

11. Before I started on a new project at Foundstone, I was usually given a list of the desired features. I was not told what algorithms to use or how to make those features actually come into existence in the software. I was expected to figure out, based on my prior knowledge and experience, how to make the computer program do what was expected.

12. One of the features on which I worked was the FoundScan web portal (aka Experience). When I started I wrote the web interface to scan management in about three weeks. It was a good application, but there was nothing I did which was not the implementation of generally known computer programming practices. Moreover, the NTO toolkit does not have any web portal.

13. After that I helped with various tasks related to fixing and/or upgrading other parts of software programs at Foundstone. This included the creation of VulnTrak, a program which maintains records of vulnerabilities reported in client computer systems. It took me about a month to write the VulnTrak computer program. Once again, while it is a good implementation, there are no algorithms or methods used in that computer program which, in my experience, are not already generally known to good programmers with experience in database programming.

14. I was never asked to participate in the preparation of any application for a patent while I was at Foundstone. I have never been advised that I am named as the "inventor" on any patent application filed by Foundstone.

FOUNDSTONE'S ALLEGED TRADE SECRETS

15. I have read the Declaration of Stuart McClure submitted in support of this OSC re: Preliminary Injunction. From review of that document I am unable to determine any specific method, algorithmortechnique which is being referred to by Foundstone as a trade secret, with the possible exception of Foundscore. However, NTO isn't doing anything like Foundscore.

16. As stated above, the work which I did while I was employed by Foundstone was good programming, but the methods, techniques and algorithms which I used were the same as those I would expect would be used by any other excellent experienced programmer.

17. The McClure declaration gives the impression that Foundstone might be claiming HTML as a trade secret. The pages which display on the world wide web are mostly HTML. All

of the HTML techniques that I used at Foundstone were standard. No new ground was broken there. All of the HTML I have done at NTO is, likewise, standard programming techniques.

18. During the entire time I was employed by Foundstone I never saw any documents, source code or other material with any "trade secret" or "secret" designation.

DECISION TO LEAVE FOUNDSTONE AND JOIN NTO

19. My commute to the Foundstone office was too far. When I learned that my wife and I were going to have a baby I wanted to be closer to home so that I could assist my wife. I had told Foundstone, even during my initial interview, that I wanted to telecommute. However, I was only able to maintain the right to telecommute two days a week by threatening to quit. This made me uncomfortable with my position at Foundstone. Moreover, I found the pace of work at Foundstone to be frustrating. I would have little to do for an extended period of time while management created a document stating what features they wanted in the software. Then I would need to work frantically, for a couple of months, actually creating the program.

20. I had no contact with JD, or anyone else at NTO, regarding potential employment, until after I left Foundstone. I gave my two week notice in mid-June. Our baby was born on June 27, 2002 and my last day at Foundstone was July 1, 2002.

21. After I left my employment at Foundstone I started to look for other employment. For a period of time I tried to work from my home but I decided I needed a more steady income. I contacted JD who I knew had started NTO after he left Foundstone. He told me that he was interested but that I would need to wait until he had enough money to fund a position for me.

22. I subsequently joined NTO where I am presently the Senior Internet Software Engineer.

23. We have, as a result of this litigation, been required to spend time meeting with our legal counsel. Based on those discussions, and based on my discussions with the other employees of NTO, and based on my knowledge of computer programming and computer techniques, neither I, nor any other person at NTO, is utilizing anything which I understand to constitute a trade secret of Foundstone in the furtherance of the business of NTO.

Steven Brower, State Bar #93568
Brian Barrow, State Bar #177906
STEPHAN, ORINGHER RICHMAN & THEODORA
A Professional Corporation
535 Anton Boulevard, Suite 800
Costa Mesa, California 92626
Telephone: (714) 241-0420

Attorneys for Defendants

SUPERIOR COURT OF THE STATE OF CALIFORNIA

FOR THE COUNTY OF ORANGE

FOUNDSTONE, INC.,
Plaintiff,
vs.
NT OBJECTIVES, INC., et al.,
Defendants.

CASE NO: 02CC15350
Judge David H. Brickner
Dept. C17

DECLARATION OF MICHAEL J. MORTON IN OPPOSITION TO OSC RE: PRELIMINARY INJUNCTION

Date: October 25, 2002
Time: 3:00 p.m.
Dept: C17

I, Michael J. Morton, declare:

1. The facts stated herein are of my own personal knowledge except for those set forth on information and belief, and as to any such facts, I have a basis for believing that they are true. If called upon to testify as a witness I could and would competently do so as set forth herein.

ROLE IN THIS LITIGATION

2. I am individually named as a defendant in this matter.

3. I was employed by Plaintiff Foundstone, Inc. from January 2001 until June 2002, as more fully set forth below. I was never an officer or director of that company.

BACKGROUND IN COMPUTERS

4. I received a bachelors degree (Phi Beta Kappa) in Information and Computer Science from the University of California at Irvine in June, 1994.

5. I have been involved with computers since 1982. Prior to entering college I was fluent with several computer assembly languages (6502, Z-80, 8088, 6809) and with microcomputer architecture. I had written computer programs that dealt with: graphics and realtime video chip programming; hardware interrupts; an 80486 assembler; dabbled in TTL/CMOS logic design; designed several circuits.

6. Since before college, and throughout my career, I have been evolving a program that does ray traced graphical rendering. This is a 3-D technique where you "draw" a ray of light with sufficient detail so that the final image has shadows, shading, perspective and reflections, all generated automatically. Part of the program includes the creation and implementation of a "description language," which is similar to creating a new computer programming language and a new computer program which understands that language and causes the computer to do what has been requested.

7. I worked for Quarterdeck Corporation from June 1994 to October 1996 as a staff programmer. I learned about Windows programming and TCP/IP programming. During this employment I wrote a PPP layer to the Winsock product, which is a networking protocol related to Windows.

8. I worked for Connect3 from October 1996 to April 1997 as a software engineer. I learned database programming including ODBC and SQL.

9. I worked for Beckman (now Beckman/Coulter) from April, 1997 to February, 2000 as a senior software engineer. I did more database programming (ODBC, Oracle and SQL Server), more networking (TCP/IP) and user interface Windows programming. I learned COM and ATL. All of this was done for a DNA Sequencer product referred to as the CEq-2000.

10. I worked for HiHo.com from February 2000 to December 2000 as a senior software engineer. I learned HTML, DHTML, ASP, WinInet, MTS and ActiveX, among others. The application we were working on was a distributed enterprise web application with a SQLServer

database. This job, in my mind, represented a culmination of my prior experience with various computer technologies and added several new ones to my repetoire (all having to do with web programming.)

EMPLOYMENT BY FOUNDSTONE

11. I was happy with my employment at HiHo. However, it was a casualty of the dot com decline. On the day that it became apparent to me that HiHo was dissolving, I received a telephone call from Bernard Lee, a recruiter, who pointed me to Foundstone.

12. After a lengthy background check I began to work for Foundstone in mid-January 2001. I was a senior software engineer for the first 3 months then chief software engineer after that.

13. Before I started on a new project at Foundstone, I was usually given a list of the desired features. I was not told what algorithms to use or how to make those features actually come into existence in the software. I was expected to figure out, based on my prior knowledge and experience, how to make the computer program do what was expected.

14. While employed by Foundstone I designed and implemented a reporting mechanism which I am informed and believe they are still using. Foundstone had previously obtained this functionality by using a software program from an outside vendor, but Foundstone wanted to save money. Substantial portions of the software I wrote for Foundstone were based upon work I had done while working for Beckman. All of the graphics techniques I used in the reporting mechanism were previously present in my raytracer software (see ¶ 6). Although I showed the product to other people in the company and discussed features and functions with them, the design of the algorithms and the coding of the software was done solely by me, or by people working to implement my instructions. While the reporting mechanism is a good computer program, and while it provides the functionality to support that piece of Foundstone's business, I am unaware of any feature in that software package that is exclusive to Foundstone. Moreover, the Fire & Water software, to be released by NTO, does not include the reporting mechanism software which was created while I was at Foundstone.

15. While employed by Foundstone I designed and implemented the FASL (Foundstone Attack Scripting Language) that I am informed and believe they are also still using as part of their

vulnerability assessment. I had previous experience writing computer languages such as the 80486 assembler which I had written before I worked at Foundstone. The graphics capabilities in that software were substantially based on my raytracer software (see ¶ 6). Although I showed the product to other people in the company and discussed features and functions with them, the design of the algorithms and the coding of the software was done solely by me, or by people working to implement my instructions. While it is a good software package, and while it provides the functionality to support that piece of Foundstone's business, I am unaware of any feature in that software package that is exclusive to Foundstone. Moreover, the Fire & Water software, to be released by NTO, does not include anything like FASL.

FOUNDSTONE'S ALLEGED TRADE SECRETS

16. I have read the Declaration of Stuart McClure submitted in support of this OSC re: Preliminary Injunction. From review of that document I am unable to determine any specific method, algorithm or technique which is being referred to by Foundstone as a trade secret.

17. To the extent that I am generally familiar with the graphical techniques which exist in the Foundstone software, because they exist in software I worked on while I was at Foundstone, it is my testimony that all of the graphical techniques that Foundstone may be claiming as proprietary are documented in public domain materials. Specifically, to my recollection, they can all be found in a book called "Computer Graphics Principles and Practice" by Foley, VanDam, Feiner and Hughes.

18. To the extent that I am generally familiar with the mathematical techniques which exist in the Foundstone software, because they exist in software I worked on while I was at Foundstone, it is my testimony that all of the mathematical techniques that Foundstone may be claiming as proprietary are documented in public domain materials, such as a standard linear algebra book. The cubic spline interpolation was done by copying code samples in a book called "Numerical Recipes" by William H. Press, Brian P. Flannery, Saul A. Teukolsky and William T. Vetterling.

19. The McClure declaration gives the impression that Foundstone might be claiming HTML as a trade secret. The pages which display on the world wide web are mostly HTML. All

of the HTML techniques that I used at Foundstone were standard. No new ground was broken there. All of the HTML I have done at NTO is, likewise, standard programming techniques.

DECISION TO LEAVE FOUNDSTONE AND JOIN NTO

20. JD Glaser and I became friends while we were both working at Foundstone. Prior to giving notice at Foundstone I asked JD whether I could work for his company. He advised me that under his Employment Agreement with Foundstone he could not have any such discussion with me while I was working for Foundstone.

21. Therefore, I gave my two week notice to Foundstone. My last day with Foundstone was June 28, 2002.

22. I subsequently applied for a job with NTO. I am presently the Senior Software Engineer for NTO.

23. We have, as a result of this litigation, been required to spend time meeting with our legal counsel. Based on those discussions, and based on my discussions with the other employees of NTO, and based on my knowledge of computer programming and computer techniques, neither I, nor any other person at NTO, is utilizing anything which I understand to constitute a trade secret of Foundstone in the furtherance of the business of NTO.

I declare, under penalty of perjury, that the foregoing is true and correct and that this declaration is executed on October 23, 2002, under the laws of the State of California.

Michael J. Morton

Steven Brower, State Bar #93568
Brian Barrow, State Bar #177906
STEPHAN, ORINGHER RICHMAN & THEODORA
A Professional Corporation
535 Anton Boulevard, Suite 800
Costa Mesa, California 92626
Telephone: (714) 241-0420

Attorneys for Defendants

SUPERIOR COURT OF THE STATE OF CALIFORNIA

FOR THE COUNTY OF ORANGE

FOUNDSTONE, INC.,

Plaintiff,

vs.

NT OBJECTIVES, INC., et al.,

Defendants.

CASE NO: 02CC15350
Judge David H. Brickner
Dept. C17

DECLARATION OF ERIK CASO IN OPPOSITION TO OSC RE: PRELIMINARY INJUNCTION

Date: October 25, 2002
Time: 3:00 p.m.
Dept: C17

I, Erik Caso, declare:

1. The facts stated herein are of my own personal knowledge except for those set forth on information and belief, and as to any such facts, I have a basis for believing that they are true. If called upon to testify as a witness I could and would competently do so as set forth herein.

ROLE IN THIS LITIGATION

2. I am individually named as a defendant in this matter.

3. I was employed by Plaintiff Foundstone, Inc. from May 2001 until June 2002, as more fully set forth below. I was never an officer or director of that company.

BACKGROUND IN COMPUTER BUSINESS

4. I obtained my bachelors degree in Business from California Polytechnic State University, San Luis Obisbo in June, 1997. My coursework included web design/authoring and using complex modeling software to create statistical forecasting models.

5. From April 1998 to March 2000, I was employed by The Boeing Company as a Business Analyst, using software for financial modeling of return on investment and other business planning.

6. From March 2000 to February 2001, I was employed by Epoch Internet as a Product Manager. I designed software based products for a large Internet Service Provider (ISP) including email system and real-time provisioning systems for an online portal.

EMPLOYMENT BY FOUNDSTONE

7. A colleague of mine from a prior job referred me to Foundstone. I liked the idea of working for a small company which was known for doing good work.

8. I began to work for Foundstone on or about May 15, 2001. My title was Product Manager. My supervisor was Dave Cole, although I primarily reported to Stuart McClure.

9. I was involved in the design and execution of the FoundScan technology. That is, I worked on deciding what features should be included in the Foundscan software in order to provide the features which Foundstone clients either requested or which Foundstone believed would increase customer demand for Foundstone services. I was also responsible for Foundstone's beta testing software program, sales assistance, development of marketing related materials, ensuring Quality Assurance for software, and overall management of FoundScan related objectives.

10. My job responsibilities at Foundstone did not include any computer programming. Although I have worked with computer companies I have not actually done any computer programming.

11. I was never asked to participate in the preparation of any application for a patent while I was at Foundstone. I have never been advised that I am named as the "inventor" on any patent application filed by Foundstone.

DIFFERENTIATING NTO AND FOUNDSTONE

12. As the person responsible for the overall business strategy of NTO I have been asked to explain the difference between Foundstone and NTO. Attached hereto as Exhibit "A," and incorporated herein by reference, is a document I prepared that provides a detailed description of the features and functions which will be included in the "Fire & Water" tools which NTO intends to distribute. Attached hereto as Exhibit "B," and incorporated herein by reference, is a document I prepared that provides my description of Foundscan, the software package utilized by plaintiff. As will be readily apparent, it is materially different from anything being done by the "Fire & Water" tool kit. Attached hereto as Exhibit "C," and incorporated herein by reference, is a comparison chart which I prepared showing features of FoundScan in comparison with those in the "Fire & Water" package. That document also shows features that are present in other software programs produced by entities who are not parties to this litigation.

FOUNDSTONE'S ALLEGED TRADE SECRETS

13. I have read the Declaration of Stuart McClure submitted in support of this OSC re: Preliminary Injunction. From review of that document I am unable to determine any specific method, algorithm or technique which is being referred to by Foundstone as a trade secret.

14. Although they are not clearly identified by the McClure Declaration, there are two databases which I believe might constitute the type of proprietary information to which Stuart McClure intends to refer. One of those might be referred to as the OS Fingerprint file. This would be the collection of information which is used by Foundstone to do Operating System identification. While I am informed that the technique of OS identification is not proprietary to Foundstone, their database of this information is extensive and might normally qualify as proprietary. The second would be the database of vulnerabilities. I am told that such a listing is actually a matter of public knowledge, but I believe this is something to which Stuart McClure was referring in this Declaration.

15. These two databases (OS Fingerprint and Vulnerabilities) are part of the FoundScan suite of security programs. During the time that I was employed by Foundstone, the OS Fingerprint database was maintained in an "unencrypted" form. That is, anyone who had access to the computer program could read and/or print the entire contents of the OS Fingerprint database.

16. As a regular part of my job at Foundstone, I was involved in sending "evaluation copies" of our software to the trade press so that they would review our products and provide us with coverage. When dealing with the press there are never any confidentiality agreements requested or obtained because the very purpose of sending the copies to the press is so that they will make public comments about your services and software.

17. I specifically recall that copies of the Foundstone software (and, therefore, full ability to view and/or print the contents of the databases) was provided to: John Taschek and Jim Rapoza of eWeek Labs; Andrew Conry-Murray of Network Magazine; Jane Parkhouse of SC Magazine; Mandy Andress of InforWorld; and, Konstantinos Karagiannis of PC Magazine. Each of those people, and their publications, had full and unfettered ability to view and/or print the entire contents of the Foundstone OS Fingerprint database.

18. Notwithstanding the potential public disclosure of this database, I want to be sure it is clear that NTO is not including any OS Identification in the "Fire & Water" package. Moreover, NTO does not have any copy of the Foundscan OS Identification database.

DECISION TO LEAVE FOUNDSTONE AND JOIN NTO

19. I left Foundstone because of its lack of interest in employee satisfaction and poor management. When I advised Dave Cole and Stuart McClure that I was quitting they asked if there was any job at the company, which they could give me, which would make me stay. When I turned them down they told me that I could return to Foundstone at any time in the future.

20. I began to look for a new job about a month before I left Foundstone. This was about a month after JD Glaser left Foundstone. I contacted him to see what he was doing since we had been friends while he was working for Foundstone. As a business person, rather than a technical person, I told him that I had some ideas about how to structure a successful organization.

21. I subsequently joined NTO where I am presently the Director, Product Strategy. My role is to develop and implement our overall product and business strategy.

22. We have, as a result of this litigation, been required to spend time meeting with our legal counsel. Based on those discussions, and based on my discussions with the other employees of NTO, and based on my knowledge of computer programming and computer techniques, neither

I, nor any other person at NTO, is utilizing anything which I understand to constitute a trade secret of Foundstone in the furtherance of the business of NTO.

I declare, under penalty of perjury, that the foregoing is true and correct and that this declaration is executed on October 23, 2002, under the laws of the State of California.

Erik Caso

EXHIBIT 'B'

What is FoundScan?

FoundScan is an advanced, network based "Vulnerability Management Technology" that is designed to identify all network based assets, their corresponding vulnerabilities, and provide a comprehensive system in which to manage the elimination of those vulnerabilities. FoundScan is deployed in a single machine or multi-machine architecture, and may be run as a distributed system where various servers are in separate locations. Deployment requires a FoundScan Consultant to be onsite performing the installation and setup, ranging from one to two days work. All FoundScan data is kept in a Microsoft SQL database, and requires the installation of this and other third party software in order to operate (the web portal requires Microsoft IIS to operate). All machines in a FoundScan system must be properly configured and maintained in order to operate properly; there is extensive work that must be performed and maintained in order to ensure proper communication between the machines in a FoundScan system or they will fail to work entirely.

Components and features of the "FoundScan Enterprise Vulnerability Management System":

- Discovery Engine - identifies, enumerates and classifies each host on a network. Classification can include the operating system, open ports, banners, hostname, and even device type (i.e. routers, firewalls, servers and 802.11 wireless devices.)
- Vulnerability Engine - includes the vulnerability checks that are performed by FoundScan and a database of all check related information (i.e. descriptions, recommendations, CVE numbers and more). FoundScan currently checks for more than 530 vulnerabilities, and grows in number on a weekly basis. The checks are preformed using a scripting language developed in-house, called FASL (FoundScan Attack Scripting Language).
- Web Portal - a comprehensive, interactive web portal that allows for complete management of the FoundScan features. Governed by rule-based access, each user has an account with discreet permissions as to how they may interact with the system; account types include Administrator, Full and View Users. The portal allows for account creation/management, viewing of scan reports, data search functions, and remediation management control.
- Remediation Engine - embedded in the Web Portal, the remediation management component, dubbed VulnTrak, allows for work flow management of vulnerabilities discovered by FoundScan. Akin to a trouble ticket system, VulnTrak allows for an Administrator to assign vulnerabilities to users, who are then alerted to the ticket and must go and fix the security issue. Once completed, they may verify and close the ticket, returning it to the Administrator for archiving.
- Numerous ancillary features, including email alerting, iDefense security reports (provided by a third party), and much more.

FoundScan is designed, marketed and sold as an enterprise security solution used to identify vulnerabilities in a corporate network. It is not currently designed to handle numerous, non-recurring projects related to network management, as the investment in such work is much greater than the resulting output (i.e. an administrator that needs to create and run numerous discreet network tests would not use FoundScan due to the amount of time it takes to log into the system, create the proper scan(s), run them, log into a web portal account, select the scan, view the information and then act.) FoundScan is designed to run as a network service, where an administrator creates recurring assessment scans, or jobs, which are run on a schedule. These jobs are then reviewed on a recurring basis and used to determine the security posture of an organization or network. Each job is configured to be automatically distributed to unlimited individuals, based on whether they have access to the job or not. FoundScan may be purchased as a licensed technology or a managed service.

Feature comparison for Fire & Water and FoundScan

The above general descriptions of each product attempt to clearly indicate the intended design of each product. With that considered, there are several features and capabilities common to both technologies, as well as many other competing technologies offered by other software vendors. While these features/capabilities may be shared, it is important to note that in not a single case is there a shared feature that is a proprietary technology of either company; various implementations and variations of these features/capabilities may be readily found in numerous software products and have been discussed and available in public forums long before the founding of either Foundstone (founded late 1999) or NT OBJECTives (founded mid-1997).

For a feature breakdown please see the Fire & Water - FoundScan Feature Comparison document.

EXHIBIT 'C'

FIRE & WATER - FOUNDSCAN FEATURE COMPARISON*

FEATURE	FIRE&WATER	FOUNDSCAN	EXISTING TOOLS DOING SAME THING
General Assessment			
TCP port scanning	YES	(YES	nmap, superscan, and numerous others
UDP port scanning	NO	YES	nmap, superscan, and numerous others
Banner grabbing	YES	YES	nmap, superscan, and numerous others
Service enumeration	NO	YES	nmap
Operating system identification	NO	YES	nmap, xprobe, queso
Hostname resolution	YES	YES	nmap, nslookup, numerous others
ICMP tracerouting	YES	YES	traceroute, visualroute, cheops
TCP tracerouting	NO	YES	firewalk, LAN MapShot
Firewall detection	NO	YES	cheops, LAN MapShot
Router detection	NO	YES	ISS, nessus, LAN MapShot, InFlow, many more
Database detection	NO	YES	AppDetective, ISS
Wireless device detection	NO	YES	ISS, nessus
General Defense			
ISAPI filter	YES	NO	URLScan
Attack signature recognition-based web server defense	YES	NO	Snort
Vulnerability Checking			
Network level checks	NO	YES	nessus, ISS, cybercop, whisker, retina + more
Operating system checks	NO	YES	nessus, ISS, cybercop, many more
Remote windows registry checks	NO	YES	nessus, ISS, cybercop, many more
Web server checks	YES	YES	nessus, ISS, cybercop, whisker, retina
[Web] application checks	NO	YES	owasp tools, WhArsenal, whisker, retina
Wireless device checks	NO	YES	nessus, ISS
Vulnerability risk rating	NO	YES	nessus, CyberCop, ISS, Shadow, many more
Custom Scripting language for vulnerability checks	NO	YES	nessus (nasl)
Interpreter for parsing and executing the scripting language	NO	YES	nessus (nasl)
Data Storage			
Enterprise database usage (MS SQL)	NO	YES	
Data Encryption	NO	YES	
Segregation of data by customer	NO	YES	
Segregation of data by user account	NO	YES	
Online Data Presentation and Management			
Web based portal with access control	NO	YES	
Online account management	NO	YES	
Integrated remediation management/workflow system (VulnTrak)	NO	YES	
Query based data searching (database)	NO	YES	
Scan scheduler	NO	YES	Retina, ISS, many more
Automated scan instantiation (starts scans)	NO	YES	Retina, ISS, many more
Automated/configurable web based alerting	NO	YES	
Automated/configurable email Alerting	NO	YES	Retina
Multi-tier account creation for granular access control	NO	YES	
Interface for complete system management	NO	YES	ISS, Retina, CyberCop, Shadow, many more
Secure communication over SSL (encrypted traffic)	NO	YES	
Includes third party security Intelligence reports	NO	YES	
Reporting			
Summary level reporting	YES	YES	ISS, CyberCop, nessus, many other
Detailed reporting on a feature basis	NO	YES	nessus, ISS
Network mapping - host level	YES	YES	cheops, CyberCop, LAN MapShot, InFlow, many more
Network mapping - firewall level	NO	YES	cheops, LAN MapShot, InFlow, many more
Network mapping - router level	NO	YES	cheops, LAN MapShot, InFlow, many more
Network mapping - dual homed devices	NO	YES	cheops, LAN MapShot, InFlow, many more
Data trending (plot historic data points on graph)	YES	YES	
Miscellaneous			
Runs only from the command line (i.e. the "C:\>" prompt)	YES	NO	nessus, nmap

Runs as a network service	NO	YES	ISS
Distributed system- multiple servers geographically based	NO	YES	ISS
Able to run numerous simultaneous scans	NO	YES	ISS, CyberCop, many more
Security scoring to indicate risk posture	NO	YES	
Available as a managed service	NO	YES	ISS
Available as software	YES	YES	All
No installation required, just copy files to desired directory	YES	NO	nmap
Requires a trained consultant for configuration and installation	NO	YES	
Requires system management to ensure operation	NO	YES	
Auto-update feature to auto download system updates	NO	YES	ISS, CyberCop, Retina, many more
Requires third party enterprise software to operate	NO	YES	
Marketing & Sales			
Target market	Security and systems administration professionals looking for lightweight tools to perform discrete tasks (such as finding web servers).	Large enterprises that require comprehensive "vulnerability management" solution.	
Price	Free, or single user license may be purchased for $150.	Published pricing begins at $30,000. Individual sales generally run $100,000-700,000 for technology licensing.	

*NOTE - This feature list comprises all known FoundScan and Fire & Water features

Steven Brower, State Bar #93568
Brian Barrow, State Bar #177906
STEPHAN, ORINGHER RICHMAN & THEODORA
A Professional Corporation
535 Anton Boulevard, Suite 800
Costa Mesa, California 92626
Telephone: (714) 241-0420

Attorneys for Defendants

SUPERIOR COURT OF THE STATE OF CALIFORNIA

FOR THE COUNTY OF ORANGE

FOUNDSTONE, INC., Plaintiff, vs. NT OBJECTIVES, INC., et al., Defendants.	CASE NO: 02CC15350 Judge David H. Brickner Dept. C17 **DECLARATION OF JASSEN D. GLASER IN OPPOSITION TO OSC RE: PRELIMINARY INJUNCTION** Date: October 25, 2002 Time: 3:00 p.m. Dept: C17

I, Jassen D. Glaser, declare:

1. I am a defendant in this action. The facts stated herein are of my own personal knowledge except for those set forth on information and belief, and as to any such facts, I have a basis for believing that they are true. If called upon to testify as a witness I could and would competently do so as set forth herein.

ROLE IN THIS LITIGATION

2. I am individually named as a defendant in this matter. I am also the president, director and majority shareholder of defendant NT Objectives, Inc. ("NTO").

3. I was previously employed by Plaintiff Foundstone, Inc. for approximately two years as more fully outlined below. I was never an officer or director of Foundstone. That is, while my title was "Director of Engineering" I was never a director in the legal sense of service on the Board

NTO. I told him that I wanted to develop one of the products which I had previously sold to Foundstone, VisualLast, which Foundstone was obligated to return to me under oral agreement. I even asked him to invest $250,000 in NTO. In fact, I have several emails from Jeanne Miller-Romero, the inhouse legal counsel at Foundstone, as late as 3 months after I resigned from Foundstone, saying that she is working on getting an agreement approved to return that software to me. However, I ultimately realized that Foundstone was not going to proceed with the oral agreement.

FOUNDSTONE'S ALLEGED TRADE SECRETS

26. I have read the Declaration of Stuart McClure submitted in support of this OSC re: Preliminary Injunction. From review of that document I am unable to determine any specific method, algorithm or technique which is being referred to by Foundstone as a trade secret.

27. As stated above, the work which I did while I was employed by Foundstone was good programming, but the methods, techniques and algorithms which I used were the same as those I would expect would be used by any other excellent experienced programmer.

28. The McClure declaration tries to give the impression that there are real technology secrets at Foundstone. I would, almost without exception, disagree, as more fully set forth below.

29. As differentiated from the technical area, I would agree that Foundstone probably has legitimate trade secrets regarding certain aspects of their customer relationships, their pricing strategy, and their financial affairs. However, those items are simply not implicated here. We are offering a different product and/or service to a different market. Foundstone's minimum service packages are over $30,000 and it was my impression that they really were not interested in packages of less than $50,000. NTO's "Fire & Water" package is free to individuals and $125 for organizations. It is not just a matter of degree. We do <u>not</u> offer a service which replaces the service offered by Foundstone. While we would be willing to do business with the same companies which are customers of Foundstone, any resulting business we received would be in addition to, not instead of, any service which those companies were obtaining from Foundstone.

30. A simple example might help to explain the material difference between the business of Foundstone and the business of NTO. The tools which NTO intends to provide as part of "Fire &

Water" are solely for stand-along use. In contrast, <u>none</u> of Foundscan can be used without connecting the software to an Enterprise SQL database (a separate special purpose computer with special purpose software).

31. Foundstone <u>does</u> have real competitors. Those companies would include Qualys, Guardent and Vigilante. Some of those companies are much larger than Foundstone and claim to offer many more capabilities than Foundstone. Those companies already claim to have the technology and the ability to do what is done by Foundstone. To thc extent they do not have such capabilities it is because they have not undertaken the expense or the effort to write the software, but it is <u>not</u> because there are "secrets" that they don't know or can't obtain from public sources.

32. We have, as a result of this litigation, been required to spend time meeting with our legal counsel. Based on those discussions, and based on my discussions with the other employees of NTO, and based on my knowledge of computer programming and computer techniques, neither I, nor any other person at NTO, is utilizing anything which I understand to constitute a trade secret of Foundstone in the furtherance of the business of NTO.

33. For every technology employed by Foundstone I can find and reference a pre-existing public source for that technology. This would specifically include, but not be limited to, clear instruction on how to do network topology maps. To the best of my knowledge, during the time I was employed by Foundstone we didn't invent any new algorithms or technological advancements.

DECISION TO LEAVE FOUNDSTONE

34. After almost two years at Foundstone I realized that I did not have any real future or potential for advancement at Foundstone. I decided to resume working for myself at NTO. I advised Stuart McClure that I would be resigning. My last day at Foundstone was May 3, 2002. In fact, on that date, I "gave" a brand new software tool to Foundstone as a departing present. I am informed and believe that the software tool has been utilized by Foundstone to enhance its marketing efforts and is currently downloadable from their website.

35. I am aware that under my employment agreement with Foundstone I am not allowed to solicit any Foundstone employee to work for NTO. Whether or not such term is legally valid I have fully complied with the spirit of the requirement. I have never contacted any employee of

Foundstone to suggest that they work for me and/or NTO. However, our software development group was relatively small and I was friends with many people at the company. When people who are employees of Foundstone have contacted me to find out whether they can work for NTO I have reminded them of the non-solicitation clause and I have told them that I can't make any agreements with them while they are employees of Foundstone.

ADDITIONAL ITEMS OF INFORMATION

36. The Declaration of Stuart McClure (¶ 30-32) refers to a presentation which I made at the Black Hat conference on July 29, 2002. It is phrased in a strange way because Mr. McClure was not actually there during my presentation. He is just reporting what he was told by someone else. Significantly, the presentations, including the one I gave with Michael Morton, are videotaped. After this litigation started we ordered a videotape of the presentation which our counsel will make available for review, by the Court, upon request. Within the past week I have reviewed everything which was said by me, and by Mr. Morton, on that videotape. There is nothing which we said which reveals or refers to any trade secrets of Foundstone.

37. In the interest of clarity, I will address one specific hearsay statement by Mr. McClure. He says in his Declaration (¶ 31) that one of the functions demonstrated at the Black Hat conference, which "mimicked the proprietary functions" of FoundScan was "determining and graphically mapping a computer network in a manner nearly identical to FoundScan." Attached hereto as Exhibit "D" is the graphical mapping of a computer network as performed by FoundScan. This example is intentionally publicly disclosed by FoundStone on their website. Attached hereto as Exhibit "E" is the version which is posted on the NTO website and which was shown at the Black Hat conference which Mr. McClure describes.

38. In the Declaration of Stuart McClure (¶ 32) he says that it would have been impossible for us to create the Fire & Water toolkit in three months. I would note the following. First, while we announced the toolkit at that time, it clearly wasn't finished. It was more than two months later that we first said we would be ready to make that tool available. Second, while I was at Foundscan I saw a PowerPoint presentation, done by Mr. McClure, which showed that my development teams were capable of preparing computer code at impressive rates, not because of any technology at

Foundscan, but because of my abilities and those of my staff. Third, as set forth in the declaration of Michael Morton, the various graphical mapping capabilities which exist in the Foundscan software are almost exclusively the byproduct of Mr. Morton's prior work on his rayscan project or are otherwise disclosed in the books to which he references. Fourth, had Mr. McClure been present at the presentation he is describing to the Court he would have known that the graphical displays which NTO intends to distribute do not mimick the proprietary functions of Foundscan.

39. The Declaration of Stuart McClure makes reference to Operating System Identification. We have elected not to provide the court with a mountain of paper showing how much of the technology, generally referred to by Foundstone, is in the public domain. However, in the interest of clarity, attached hereto as Exhibit "F" is Version 2.5 of the paper "ICMP Usage in Scanning" by Ofir Arkin, the founder of the Sys-Security Group. Ofir Arkin does not have any affiliation with Foundstone or with NTO. This paper can be obtained as a free download, without any registration, payment or agreement to any conditions. This document discloses, to those who take the time to understand the document, how to do what is referred to by Mr. McClure as OS Identification. I know that this document forms the basis for the OS Identification in the FoundScan product because various versions of this document were used by my development team in order to implement OS Identification at Foundstone.

40. Further, attached hereto as Exhibit "G" is the home page from www.sys-security.com, the organization operated by Ofir Arkin. At the bottom of the page it refers to Xprobe2 "an active operating system fingerprinting tool with a different approach to operating system fingerprinting. Xprobe2 rely on fuzzy signature matching, probablistic guesses, multiple matches simultaneously, and a signature database." It also indicates that you can download the source code, for free, from this site. We are not arguing that this is the exact same thing as what Foundstone is offering. In fact, in some ways it may even be more sophisticated than Foundstone. Our concern is that when Foundstone claims to "own" OS Identification, or any of their other alleged trade secrets, without defining what they have which is different or special (i.e. - which might qualify as a trade secret), we can't show that the specific item: a) is not, in fact, a secret; or b) is not, in fact, used by NTO.

41. While we reserve the right to use any non-proprietary technology in the future, including OS Identification, it should be clearly noted that the "Fire & Water" package does not include any OS identification and has never been intended to include such capability.

I declare, under penalty of perjury, that the foregoing is true and correct and that this declaration is executed on October 23, 2002, under the laws of the State of California.

Jassen D. Glaser

Steven Brower (State Bar No. 93568)
Brian P. Barrow (State Bar No. 177906)
STEPHAN, ORINGHER, RICHMAN & THEODORA, P.C.
535 Anton Boulevard, Suite 800
Costa Mesa, California 92626-1902
Telephone (714) 241-0420
Telecopier (714) 241-0622

Attorneys for Defendants
NT OBJECTIVES, INC., J.D. GLASSER,
MICHAEL MORTON, ERIK CASO and
DAN KUYKENDALL

SUPERIOR COURT OF THE STATE OF CALIFORNIA

FOR THE COUNTY OF ORANGE

FOUNDSTONE, INC., a Delaware corporation

Plaintiff,

v.

NT OBJECTIVES, INC., a California corporation; J.D. GLASER, an individual; MICHAEL MORTON, an individual; ERIC CASO, an individual; DAN KUYKENDALL, an individual; and DOES 1 through 50, Inclusive.

Defendants.

Case No. 02CC15350

ASSIGNED FOR ALL PURPOSES TO:
HONORABLE DAVID H. BRICKNER
DEPT: C17

MEMORANDUM OF POINTS AND AUTHORITIES IN SUPPORT OF DEFENDANTS' OPPOSITION TO PLAINTIFF'S OSC RE: PRELIMINARY INJUNCTION

HEARING DATE: October 25, 2002
TIME: 3:00 p.m.
DEPT: C17
TRIAL DATE: None

COMPLAINT FILED: October 2, 2002

MEMORANDUM OF POINTS AND AUTHORITIES

I. INTRODUCTION

Plaintiff Foundstone, Inc. seeks to enjoin its former employees from releasing "Fire & Water Toolkit," a computer software product that Foundstone has never seen, but which allegedly incorporates misappropriated trade secrets. Foundstone fails to identify the supposedly stolen trade secrets or specify how they are supposedly being used in defendants' product, instead relying on an allegation that defendants could only have developed their product by virtue of their prior employment at Foundstone.

allegation its "algorithms, methods, and databases" are trade secrets thus does not describe which, if any, of the components of "Fire and Water Toolkit" were supposedly misappropriated from Foundstone. Without such descriptive information, Foundstone fails to demonstrate any likelihood of success on the merits or make any showing of relative interim harm.

3. Foundstone Also Refuses To Provide Further Specificity Of Its Alleged Trade Secrets As Required By the Discovery Act.

Code of Civil Procedure section 2019, subdivision (d), states the following with regard to identification of trade secrets prior commencing discovery:

> In any action alleging the misappropriation of a trade secret . . . , before commencing discovery relating to the trade secret, the party alleging the misappropriation shall identify the trade secret with reasonable particularity subject to any orders that may be appropriate under [the Uniform Trade Secret Act].

Defendants have repeatedly asked Foundstone for such an identification, specifically explaining that Foundstone's pleadings contain an insufficient description of the alleged trade secrets. Tellingly, Foundstone has so far refused to provide defendants with the requested specificity, despite previously seeking leave to commence the discovery process.

D. Foundstone's "Software, Methods, and Algorithms" Are Not Protectible Trade Secrets.

Even if the generic "software, methods, and algorithms" is considered a sufficient description of a trade secret, Foundstone nevertheless fails to provide evidence demonstrating that such material is entitled to trade secret protection. Foundstone thus fails to show that its claimed "software, methods, and algorithms" are actually trade secrets.

"The test for trade secrets is whether the matter sought to be protected is information (1) which is valuable because it is unknown to others and (2) which the owner has attempted to keep secret." (*Schlage Lock Company v. Whyte* (2002) 101 Cal.App.4th 1443, 1453, citing *Abba Rubber Co. v. Seaquist* (1991) 235 Cal.App.3d 1, 18.) Foundstone offers no evidence satisfying either of these requirements.

1. Foundstone's Alleged Trade Secrets and Software Products Are Based on Information Available In the Public Domain.

The United States Supreme Court has stated that information that is public knowledge or that is generally known in an industry cannot be a trade secret. (*Ruckelshaus v. Monsanto Co.* (1984) 467 U.S. 986, 1002.)

As shown in the Declarations submitted in Opposition, Foundstone did not actually invent anything. To the extent that defendants are able to identify alleged trade secrets the Declarations show that such materials were originally created by sources outside of Foundstone, are known to those who are experienced in the industry, or have been deliberately disclosed to the public by Foundstone.

2. Foundstone Disclosed Any Trade Secrets Contained In Its Software When It Provided Unprotected Marketing Versions to the Press.

Foundstone cannot validly claim that its various databases upon which FoundScan operates have been the subject of efforts to prevent the loss of secrecy. As explained in the declaration of defendant Erik Caso, Foundstone repeatedly provided members of the press with copies of its software products that were vulnerable to disclosure. Foundstone never insisted that the members of the press to whom the software was delivered execute nondisclosure agreements or otherwise promise not to publicize its alleged trade secrets. As such, members of the press who received the software were free to, even encouraged to, view the very components of the software (i.e. databases, etc) that Foundstone now contends are trade secrets they have attempted to keep confidential. Foundstone's own acts of disclosing information to outsiders demonstrates that the alleged trade secrets no longer enjoy the secrecy that Foundstone claims in this lawsuit.

E. Foundstone Has No Evidence of any Actual or Threatened Misappropriation of "Software, Methods, and Algorithms."

This court may only enjoin "actual or threatened misappropriation" of a trade secret. (Civ. Code, § 3426.1, subd. (a).) "Misappropriation" is defined in this context as improper acquisition of a trade secret or its nonconsensual use or disclosure. (Civ. Code, § 3426.1, subd. (b).) Foundstone is therefore burdened with making a showing that defendants are either

Create an account

Home • Topics • Downloads • Your Account • Submit News • Top List

October 25, 2002

Main Menu

- Downloads
- FAQ
- Journal
- Members List
- Messages
- News
- Photo Gallery
- Polls
- Recommend Us
- Reviews
- Search
- Stats
- Submit News
- Topics
- Top List
- Web Links

Who's Online

We have 20 guests and 1 member online

You are an anonymous user. You can register for free by clicking here

Administration

- Admin Page

Languages

Select interface language:

English

Welcome to the Kuykendall Website.

Hello, I am Dan Kuykendall (aka Seek3r). Welcome to my website. This site is intended to be for the use of the visitors and myself to discuss whatever issues are on our minds.

Legal Docs: The response to their complaint

In this article on Fool.com there is a great example of what Foundstone is doing:
'Finally, there's bluffing. This happens all the time. During the 1992 presidential campaign, Ross Perot copyrighted his likeness and convinced Dana Carvey to stop parodying him on Saturday Night Live, despite the fact that parody explicitly falls under "fair use" and they had been parodying other copyrighted and trademarked properties (such as Eddie Murphy doing "Gumby" and "Buckwheat") for years. The important thing wasn't whether Perot had a case that would hold up in court, it's that he had a lot of money to spend on lawyers, he sounded serious, and that Dana Carvey backed down.'

The fact is that we have done nothing wrong, but they have money and act serious to intimidate us. The problem for them is... We are not intimidated!

Here is the full collection of legal docs:
The new stuff:
Here are the declarations of all the defendants in the case; JD Glaser, Mike Morton and myself. Here is our Defendants Memorandum of Opposition and their final reply. Now its in the judges hands and his decision will be posted on his website by 4:30PM PST, Oct 25 2002.
The old stuff:
The Temporary Restraining order and Stuart McClure's Declaration.
Their complaint - 1-10 / 11-20 / 21-23 and the Plaintiffs Memorandum - 1-10 / 11-16

Click the read more link for *my* opinions on each point of Stuart's Declaration.

Posted by: seek3r on Thursday, October 24, 2002 - 11:46 AM PST
Read more... (9283 bytes more) comments?

Legal Case: Here comes the press

Things are moving along with the lawsuit. Friday is going to be the day that the Judge will make the decision on the Pelminary Injunction. If we win, then we will be free to release our toolkit. The press has gotten ahold of the story as well. It has shown up on the very popular Info Security Wire and on Security Administrator.

Click the Read more... link for some internal problems at Founstone, that stem from this case.

Posted by: seek3r on Tuesday, October 22, 2002 - 12:07 AM PST
Read more... (681 bytes more) comments?

Job a

- NT OB
- phpGro
- Clear R

Surve

Should

- Yes, wi
- I dunno
- No, its choice
- Is that desert pl

[Res

Co

User's

Usernan

Passwor

☐ Rer

Don't ha yet? You As a regi have son like a the comment and post with your

EXHIBIT A PAGE 1 OF 4

Create an account Home • Topics • Downloads • Your Account • Submit News • Top List October 25, 2002

Main Menu

- Downloads
- FAQ
- Journal
- Members List
- Messages
- News
- Photo Gallery
- Polls
- Recommend Us
- Reviews
- Search
- Stats
- Submit News
- Topics
- Top List
- Web Links

Who's Online

We have 20 guests and 0 members online

You are an anonymous user. You can register for free by clicking here

Administration

- Admin Page

Languages

Select interface language:

English

Legal Docs: The response to their complaint

Posted by: seek3r

In this article on Fool.com there is a great example of what Foundstone is doing:
'Finally, there's bluffing. This happens all the time. During the 1992 presidential campaign, Ross Perot copyrighted his likeness and convinced Dana Carvey to stop parodying him on Saturday Night Live, despite the fact that parody explicitly falls under "fair use" and they had been parodying other copyrighted and trademarked properties (such as Eddie Murphy doing "Gumby" and "Buckwheat") for years. The important thing wasn't whether Perot had a case that would hold up in court, it's that he had a lot of money to spend on lawyers, he sounded serious, and that Dana Carvey backed down.'

The fact is that we have done nothing wrong, but they have money and act serious to intimidate us. The problem for them is... We are not intimidated!

Here is the full collection of legal docs:
The new stuff:
Here are the declarations of all the defendants in the case; JD Glaser, Mike Murton and myself. Here is our Defendants Memorandum of Opposition and their final reply. Now its in the judges hands and his decision will be posted on his website by 4:30PM PST, Oct 25 2002.
The old stuff:
The Temporary Restraining order and Stuart McClure's Declaration.
Their complaint - 1-10 / 11-20 / 21-23 and the Plaintiffs Memorandum - 1-10 / 11-16

Click the read more link for *my* opinions on each point of Stuart's Declaration.

My quick response to Stuart McClure's Declaration which is the only actual "facts" they have presented to the court.
These are my opinions and while I believe everything I am stating to be true, this is not a legal rebuttal. I will be posting all our legal responces as they become availible.

First lets look at what the law considers a Trade Secret:
A good explaination basicly says that "A trade secret is any information that allows you to make money because it is not generally known". Since hacker techniques are generally known and published on several websites, as well as techniques for port scanning, operating system fingerprinting and visualized network mapping from traceroute data, they cannot claim them as Trade Secrets.

Read my responses along side of his Declaration.
1) True enough
2) Notice how he makes it specific who their target audience/market is. This is not the same as NTO's which is releasing small command line tools for the general network and security admin.
3) No problem with this

Related links

- More about Legal Issues
- News by seek3r

Most read story in Legal Issues:
My Right to work

EXHIBIT A PAGE 2 OF 4

4) I question the actual costs/hours. From my calculations a single persons's full time hours for a year come out to about 2,000. To account for 80,000 would mean 20 people ($100k average annual salary/cost) working on it full time for the last two years its been in development.
Doesnt seem accurate... but its not an important point to the case... but one that he repeats several times, so I felt I should shoot it down up front.
5) For sure their source code, and databases are their own. Possibly the actual algorithm for calculating their often discussed "FoundScore". Other than that I am not personally aware of any other Trade Secrets that they have.
6) He says in this that FoundScan is the heart of Foundstone's business. This is does not make sense in terms of revenue as far as I know. Foundstone has been a very sucessful "Consulting and Training" company up until their recent release of FoundScan. I dont see how a new product that has only sold a limited number of copies has outstripped their long established consulting and training business.
7) What he is talking about is operating system "fingerprinting" for "OS Ident". This is a very common and well understood process in the industry. Yes they have their proprietary implementation, but "fingerprinting" and "OS Ident" (which we dont even do yet) is certainly not some Trade Secret to Foundstone.
8) Again, they have their own proprietary implementation, but visually mapping a network based on traceroute data is well understood and exists in many products. In Stu's own book he talks about Cheops which does this and has existed far longer than FoundScan.
9) Again, they have their own proprietary implementation and technique for automated vulnerability testing, but as far as I know all of the vulnerability checks they (and just about anyone else) do are well described on sites such as Security Focus. They then display this in a spherical map which they have examples of on their own website... how he still thinks this is a trade secret is a mystery to me. Regardless, this is not something we are even doing.
10) Again, they have their own proprietary implementation, but this is just talking about a web interface to an application. If I had stolen their code, this is a problem. Finally, I am not even doing anything even remotely simliair for NTO.
11) Again, they have their own proprietary implementation, but web crawling to inventory and hack is a well understood and documented process in the security industry.
12) Again, they have their own proprietary implementation, but most of this is just trend reporting. Stuff that tools like Microsoft Excel and Crystal Reports have been doing for years. Trending is trending is trending. The mentioned "scoring system" *is potentially* a secret algorithm, but one that they talk about the details of in the Hacking Exposed book and other forums. But I do think the exact algorithm itself is a secret... one which I dont even know. This is also not something we are doing at NTO.
13 - 17) I have no problem with these
18) This is not entirely true. I wont go into the details... but up till very recently some parts were plain text. Regardless, the general point is true enough.
19 & 20) No problem with these
21) This is not true. The documents I have seen from the purchase do not cover NTO Scanner, so the word "all" is not accurate.Whats more is that scanning is the heart of the Fire & Water toolkit. So we already had code to work from, I dont know if any of the old scanner code was actually used for the new scanner, but we had some to use none the less.
22 & 23) No problem with these
24) This is kind of lame and as far as Im aware they knew

EXHIBIT A PAGE 3 OF 4

about his desire to bring NTO back to life. Additionally NTO is *not* a direct competitor to Foundstone. We happen to be another computer security tool company... but that does not make us a direct competitor since we are not targetting their Fortune 500 market with a like product.
25 - 30) No problem with these
31) He calls the 3 functions, "proprietary functions" which means that the given functionality and not just their implementation is proprietary. I am pretty sure the only way to make a "function" proprietary is to get a Patent, which they do not have and could not get because of the vast amount of prior art. Point 1 is just a network map from traceroute data, as mentioned before. Also he calls our map nearly identical to their map. See for yourself his exageration. Number 2 seems to indicate that using a hypertext solution for reporting (aka HTML) is not allowed. For one, we dont even have such a tool... for another we almost exclusively use XML and XSLT for all of our data and reporting, AND its not a trade secret to use HTML for a report! Number 3 is again not something we even have a product to do, and its no trade secret.
32) I take great personal issue with this. The NTO teams happens to be some of the best from their own development team. I have proven to him and the entire development team on a couple occasions what I can accomplish in a short period of time. I wrote their entire web interface to scan management in about 3 weeks when I first started with the company. I also wrote their vulntrak app in about a month. The other developers at NTO are even probably even better programmers than I am. This is really just a personal insult... and its also not true. We have produced rapidly... but even then we had just barely been ready to release the products. What we showed at Blackhat was demo's and mock-ups for the most part.
33) No problem with this
34) This is so lame. First the security industry has been built on disclosure of techniques for hacking and securing. The point that their Fortune 500 customers would forgo buying a $150k software package that integrates over a hunded features and provides enterprise scalability and management, for a set of DOS PROMPT/Command Line utilities is a joke!
35) Yes, so we are not rich :-) I do have a problem with that... but thats not an issue for this ;-)
36) I have made my case against this lame point, but I will wait for him to explain Foundstone's Free tools and Stuarts own book called Hacking Exposed which explains in great detail the methods of hacking.
37 - 39) No problem with these.

In the end I hope I have explained how weak their case is. I will state again that "WE HAVE NOT STOLEN THEIR CODE OR ANYTHING THAT IS THEIR ACTUAL TRADE SECRETS". We have our own ideas, our own code and our own products. I wish Foundstone would just leave us alone, stop bluffing and trying to make us burn our money on the legal costs for defending ourselves. I encourage anyone that wants to, to email them and give them your opinions (pro or con, that is your choice).
I do hope you encourage them to back off and let us go about our business and them go back to theirs.

Dan Kuykendall

Legal Docs: The response to their complaint | Login/Create an account | 0 Comments

Threshold 0 Thread Oldest first

Comments are owned by the poster. We aren't responsible for their content.

Home · Topics · Downloads · Your Account · Submit News · Top List

EXHIBIT A PAGE 4 OF 4

Darrell L. Olson, Bar No. 77,633
Michael K. Friedland, Bar No. 157,217
Douglas T. Hudson, Bar No. 210,385
KNOBBE, MARTENS, OLSON & BEAR, LLP
2040 Main Street
Fourteenth Floor
Irvine, California 92614
(949) 760-0404 (telephone)
(949) 760-9502 (facsimile)

Attorneys for Plaintiff FOUNDSTONE, INC.

SUPERIOR COURT OF THE STATE OF CALIFORNIA

COUNTY OF ORANGE, CENTRAL JUSTICE CENTER

FOUNDSTONE, INC., a Delaware corporation, Plaintiff, v. NT Objectives, Inc., a California corporation; J.D. Glaser, an individual; Michael Morton, an individual; Eric Caso, an individual; Dan Kuykendall, an individual; and DOES 1 through 50, inclusive, Defendants.	CASE NO. 02CC1530 ASSIGNED FOR ALL PURPOSES TO: COMMISSIONER ELEANOR M. PALK DEPT. C62 **DECLARATION OF STUART MCCLURE IN SUPPORT OF PLAINTIFF'S APPLICATION FOR TEMPORARY RESTRAINING ORDER AND PRELIMINARY INJUNCTION, REQUEST FOR EXPEDITED DISCOVERY, AND REQUEST FOR FILING UNDER SEAL** HEARING DATE: Oct. 4, 2002 TIME: 1:30 p.m. DEPT: C62 TRIAL DATE: None COMPLAINT FILED: Oct. 2, 2002

I, Stuart McClure, declare:

1. I am the President and Chief Technology Officer of Foundstone, Inc., Plaintiff in the above-titled action. I have served as President and Chief Technology Officer of

Foundstone since 1999. The following is true of my own personal knowledge; and if called and sworn as a witness, I could and would testify competently thereto.

2. Foundstone specializes in computer network security and provides network security audits, education services, and security software for Fortune 500 companies, government agencies, and others worldwide.

3. Foundstone's principal product is FoundScan, a unique proprietary software system that automatically scans computer networks, tests networks for "hacking" vulnerabilities, provides detailed graphical reports and maps of tested networks and vulnerabilities, scores the security of networks, and reports on results of those tests.

4. Over the past three years, Foundstone has invested more than $4 million dollars and more than 80,000 person-hours in research, development, and testing of the proprietary FoundScan software.

5. Foundstone's proprietary FoundScan software contains a number of well-guarded algorithms, methods and databases that are highly valuable trade secrets of Foundstone. In particular, FoundScan contains trade secret software technology relating to securing corporate and government computer networks against hackers, cyber-terrorists, and electronic espionage.

6. This trade secret technology is the heart of Foundstone's business, and its value would be substantially diminished if it were to fall into the public domain. So long as the technology continues to be maintained as secret,

competitors and potential competitors would have to invest similarly large amounts of time and money to develop systems with similar capabilities to compete.

7. Foundstone's proprietary methods and databases for operating system identification include techniques for sending TCP (Transmission Control Protocol) and ICMP (Internet Control Messaging Protocol) packets to a target computer, which then responds with packets of its own. Based on the target computer's response, the operating system of the target computer is identified against a proprietary database.

8. Foundstone's proprietary methods and databases for determining and graphically mapping a computer network include methods for ICMP and TCP tracerouting to all devices, analyzing the results of the tracerouting to understand the placement of those devices on the network, and then visually displaying those devices in a three-dimensional map.

9. 'Foundstone's proprietary methods and databases for detecting network vulnerabilities and reporting them with the network map include methods for sending a series of packets to a target device to prompt a particular response. Once the response is collected, it is analyzed and compared to a series of rules that define a vulnerable device. Once identified as vulnerable, the device is then associated with a device discovered during the network mapping stage, and the result is displayed on the three-dimensional spherical map.

10. Foundstone's proprietary methods and databases for a remote administration web interface include methods for alerting a user of newly discovered vulnerabilities,

proprietary databases of vulnerabilities, tracking of discovered vulnerabilities through a workflow process of fixing them, displaying reports of found vulnerabilities, displaying threat information, displaying and controlling the status of scans, managing user and administrative roles within the web interface, and searching the proprietary database for relevant vulnerability information.

11. Foundstone's proprietary web modules for vulnerability testing of web servers include methods for "crawling" a web site for links, inventorying the technologies in place, "brute forcing" authentication mechanisms to unearth easy-to-guess passwords, guessing the names of existing but not readily linked-to files, testing for poor script input validation, and testing for source code disclosure issues.

12. Foundstone's proprietary methods and databases for reporting vulnerability results over time include methods for an objective security scoring mechanism, a breakdown of network inventory by live hosts, services open, a vulnerability network map, operating systems running, vulnerabilities found, web module analysis, and trending of these results over time.

13. Foundstone has taken extensive steps to protect the trade secrets contained in the FoundScan software, as is necessary to protect the substantial economic and competitive value of Foundstone's trade secrets. These secrets provide efficient network scanning, effective graphical mapping of networks and vulnerabilities on those networks, and understandable reports on network security.

14. Foundstone protects its software internally with passwords, "software source safes," firewalls and other network security measures.

15. Foundstone's physical facilities are protected by security measures including closed-circuit cameras, security badges, and biometric devices.

16. All of Foundstone's customers are under strict non-disclosure agreements and covenants not to reverse engineer FoundScan. Attached as Exhibit "A" to this declaration are true and correct copies of three example non-disclosure agreements of the type signed by Foundstone's customers.

17. FoundScan service customers do not have any access to the FoundScan software. These customers only interact with the software through a web interface, while the software runs from computers controlled by Foundstone.

18. The few FoundScan customers who do have access to the FoundScan software only have access to object code.

19. All of Foundstone's employees, including Defendants J.D. Glaser ("Glaser"), Michael Morton ("Morton"), Eric Caso ("Caso") and Dan Kuykendall ("Kuykendall"), signed agreements recognizing the trade secret status of its software and research. Attached as Exhibit "B" to this declaration are true and correct copies of the agreements signed by each of the above-named individuals.

20. All of Foundstone's employees, including Defendants Glaser, Morton, Caso, and Kuykendall, signed non-disclosure agreements as part of their agreements promising not to divulge Foundstone's trade secrets, and they stipulated and

Darrell L. Olson, Bar No. 77,633
Michael K. Friedland, Bar No. 157,217
Douglas T. Hudson, Bar No. 210,385
KNOBBE, MARTENS, OLSON & BEAR, LLP
2040 Main Street
Fourteenth Floor
Irvine, California 92614
(949) 760-0404 (telephone)
(949) 760-9502 (facsimile)

Attorneys for Plaintiff FOUNDSTONE, INC.

SUPERIOR COURT OF THE STATE OF CALIFORNIA

COUNTY OF ORANGE, CENTRAL JUSTICE CENTER

FOUNDSTONE, INC., a Delaware corporation,

Plaintiff,

v.

NT Objectives, Inc., a California corporation; J.D. Glaser, an individual; Michael Morton, an individual; Eric Caso, an individual; Dan Kuykendall, an individual; and DOES 1 through 50, inclusive,

Defendants.

CASE NO. 02CC1530

ASSIGNED FOR ALL PURPOSES TO:
COMMISSIONER ELEANOR M. PALK
DEPT. C62

PLAINTIFF'S MEMORANDUM OF POINTS AND AUTHORITIES IN SUPPORT OF APPLICATION FOR TEMPORARY RESTRAINING ORDER, PRELIMINARY INJUNCTION AND EXPEDITED DISCOVERY

HEARING DATE: Oct. 4, 2002
TIME: 1:30 p.m.
DEPT: C62
TRIAL DATE: None

COMPLAINT FILED: Oct. 2, 2002

secrecy." Vacco Industries, Inc. v. Van Den Berg, 5 Cal.App. 4th 34, 50 [quoting Cal. Civ. Code § 3426.1(d)].

Misappropriation of a trade secret includes "[d]isclosure or use of a trade secret of another without express or implied consent by a person who . . . (B) At the time of disclosure or use, knew or had reason to know that his or her knowledge of the trade secret was: . . . (ii) Acquired under circumstances giving rise to a duty to maintain its secrecy or limit its use; or by improper means of disclosure including breach of duty to maintain secrecy." PMC, Inc. v. Kadisha, (2000) 78 Cal.App.4th 1368, 1382-83 (quoting Cal. Civ. Code § 3426.1(b)). Because each of these elements are met here, Foundstone is likely to prevail on the merits.

1. **Foundstone Has Established the Existence of Trade Secrets**

The Uniform Trade Secrets Act defines a trade secret as "information, including a formula, pattern, compilation, program, device, method, or technique, or process that [] derives independent economic value, actual or potential, from not being generally known to the public or to other persons who can obtain economic value from its disclosure or use." Cal. Civ. Code § 3426.1(d). Schlage Lock Co. v. Whyte, 2002 Cal. App. LEXIS 4634, *13 (Cal. App. 4th Dist. Sept. 12, 2002) (affirming trade secret status of misappropriated materials).

Foundstone's technologies certainly meet this definition. The Foundstone technologies at issue are:

(1) Foundstone's proprietary methods and databases for operating system identification, which include techniques for

sending TCP (Transmission Control Protocol) and ICMP (Internet Control Messaging Protocol) packets to a target computer, which then responds with packets of its own. Based on the target computer's response, the operating system of the target computer is identified against a proprietary database of over eight hundred operating system "fingerprints." (McClure Decl., ¶7.)

(2) Foundstone's proprietary methods and databases for determining and graphically mapping a computer network, which include ICMP and TCP tracerouting to all devices, analyzing the results of the tracerouting to understand the placement of those devices on the network and then visually displaying those devices in a three-dimensional map. (Id., ¶8.)

(3) Foundstone's proprietary methods and databases for detecting network vulnerabilities and reporting them with the network map, which include sending a series of packets to a target device to prompt a particular response. Once the response is collected, it is analyzed and compared to a series of rules which define a vulnerable device. Once identified as vulnerable, the device is then associated with a device discovered during the network mapping stage, and the result is displayed on the three-dimensional spherical map. (Id., ¶9.)

(4) Foundstone's proprietary methods and databases for a remote administration web interface, which include alerting a user of newly-discovered vulnerabilities, proprietary databases of vulnerabilities, tracking of discovered vulnerabilities through a workflow process of fixing them, displaying reports of found vulnerabilities, displaying threat

information, displaying and controlling the status of scans, managing user and administrative roles within the web interface, and searching the proprietary database for relevant vulnerability information. (Id., ¶10.)

(5) Foundstone's proprietary web modules for vulnerability testing of web servers, which include "crawling" a web site for links, inventorying the technologies in place, "brute forcing" authentication mechanisms to unearth easy-to-guess passwords, guessing the names of existing but not readily linked-to files, testing for poor script input validation, and testing for source code disclosure. (Id., ¶11.)

(6) Foundstone's proprietary methods and databases for reporting vulnerability results over time including an objective security scoring mechanism, a breakdown of network inventory by live hosts, services open, a vulnerability network map, operating systems running, vulnerabilities found, web module analysis, and trending of these results over time. (Id., ¶12.)

These technologies are not generally known to the public or to those who could obtain economic value from their disclosure or use. As discussed above, Foundstone created these technologies only through investing 80,000 person-hours and $4,000,000 of research and development. (Id., ¶4.)

These technologies also derive value from their secrecy. The resulting technologies are valuable to Foundstone only because - and only for so long as - the technologies are secret. (Id., ¶6.) Foundstone is able to market its software and services effectively because it offers a product that is unique. (Id., ¶3.) So long as the technologies continue to be maintained as secrets, competitors and potential competitors would have to invest similar large amounts of time and money to develop systems with similar capabilities to compete with Foundstone. (Id., ¶6.) If the technologies were to be disclosed, entities could compete with Foundstone without having to make these investments, thereby devastating the market position Foundstone established as a result of its huge research and development investment. (Id.)

2. **Foundstone Has Taken Far-Reaching Steps to Protect the Secrecy of its Technologies**

As described above, Foundstone has taken extensive steps to protect the confidential status of its trade secrets.

Foundstone protects its software internally with passwords, firewalls, "software source safes," and other network security measures. (Id., ¶¶13-14.) Foundstone's physical facilities are protected by security measures including "biometric" security devices. (Id., ¶15)

All of Foundstone's customers are under strict non-disclosure agreements and covenants not to reverse engineer. (Id., ¶16 & Ex. A) Foundstone's service customers do not have any access to the Foundstone's software. These customers only interact with the software through a web interface while the software runs on computers controlled by Foundstone. (Id., ¶16.) The few Foundstone customers who have access to Foundstone's software only see object code. (Id., ¶17).

All of Foundstone's employees, including the individual

Defendants, signed agreements recognizing the trade secret status of Foundstone's software and research. (Id., ¶19-20 & Ex. B.) In addition, all of Foundstone's employees, including the individual Defendants, signed non-disclosure agreements promising not to divulge Foundstone's trade secrets. (Id.)

3. Defendants Indisputably Had Access to Foundstone's Trade Secrets

There can be no dispute that Defendants had access to Foundstone's trade secrets. Each of the individual Defendants was employed by Foundstone and, while at Foundstone, had key roles in developing the trade secret technologies that are at issue in this Motion. (Id., ¶29.) Defendant NTO was founded by and consists of the individual Defendants. (Id.)

4. Defendants' Release of Their Software Would Breach Their Duties to Maintain Secrecy

Each of the individual Defendants executed employment agreements that included comprehensive nondisclosure provisions. (Id., ¶19 & Ex. B.) By releasing software containing Foundstone's trade secret technologies, Defendants would certainly be breaching their obligations of confidentiality pursuant to their agreements with Foundstone.

Because the evidence establishes both that the balance of the hardships weighs dramatically in favor of Foundstone and that Foundstone is likely to prevail on the merits, this Court should temporarily restrain and preliminarily enjoin Defendants from using or disclosing Foundstone's trade secrets. Vacco Industries, Inc., 5 Cal. App. 4th 34, 53-54; Merrill Lynch, 2001 U.S. Dist. LEXIS at *14-*17.

Darrell L. Olson, Bar No. 77,633
Michael K. Friedland, Bar No. 157,217
Douglas T. Hudson, Bar No. 210,385
KNOBBE, MARTENS, OLSON & BEAR, LLP
2040 Main Street
Fourteenth Floor
Irvine, California 92614
(949) 760-0404 (telephone)
(949) 760-9502 (facsimile)

Attorneys for Plaintiff FOUNDSTONE, INC.

SUPERIOR COURT OF THE STATE OF CALIFORNIA

COUNTY OF ORANGE, CENTRAL JUSTICE CENTER

FOUNDSTONE, INC., a Delaware corporation,

Plaintiff,

v.

NT Objectives, Inc., a California corporation; J.D. Glaser, an individual; Michael Morton, an individual; Eric Caso, an individual; Dan Kuykendall, an individual; and DOES 1 through 50, inclusive,

Defendants.

CASE NO. 02CC1530

ASSIGNED FOR ALL PURPOSES TO:
HON. DAVID H. BRICKNER
DEPT. C17

PLAINTIFF'S REPLY IN SUPPORT OF MOTION FOR PRELIMINARY INJUNCTION

HEARING DATE: Oct. 25, 2002
TIME: 3:00 p.m.
DEPT: C17
TRIAL DATE: None

COMPLAINT FILED: Oct. 2, 2002

grant the preliminary injunction.").

C. The Facts Establish Likelihood Of Success

The second prong of the two-part test for injunctive relief is likelihood of success on the merits. Foundstone has clearly established that it is likely to succeed.

1. Foundstone's Specified Methods and Techniques Are Protectable Trade Secrets

In its moving papers, Foundstone submitted detailed evidence that its methods and techniques in six specific areas constituted protectable trade secrets. (Memo. at 9-11.) In response, Defendants have raised a number of arguments. All lack merit.

(a) The Trade Secrets Are Adequately Defined

Defendants argue that Foundstone has not identified its trade secrets with sufficient particularity. (Opp. at 6-8.) To make this argument, Defendants ignore the specific identification of trade secrets contained in Foundstone's moving papers and claim that Foundstone only generically stated "that Foundstone's software contains 'algorithms, methods, and databases.'" (Id. at 6.) This is simply incorrect. Foundstone's moving papers stated that its trade secrets related to six specifically described technologies. The detailed description of these technologies spanned two full pages of Foundstone's brief. (Memo. at 9-11; McClure Decl. ¶¶ 5-12.)[1]

[1] Defendants' reliance on Diodes, Inc. v. Franzen, 260 Cal.App.2d 244 (1968) is misplaced. In Diodes, the plaintiff simply referred to a "secret process." In contrast,

Defendants similarly assert that they are unable to understand the described trade secrets. (See, e.g., Glaser Decl. ¶ 26.) Such assertions are implausible, given that the Defendants themselves allege that they had critical roles in developing Foundstone's software. (See, e.g, id. ¶ 38.) The assertions are also thoroughly repudiated by Defendants' own declaration; The declarations discuss specific methods and algorithms allegedly implicated by the claimed trade secrets in great detail. (Caso Decl. ¶¶ 12-18; Glaser Decl. ¶¶ 16-19; Morton Decl. ¶¶ 16-19.)[2] See Whyte v. Schlage Lock Co., 101 Cal.App.4th 1443, 1453 (2002) (rejecting assertion that trade secrets were defined too broadly where, from responses to discovery, it was clear that former employee had "no difficulty in understanding the scope of the putative trade secret information.")

Moreover, the scope of trade secrets are specifically limited to methods developed by Foundstone. (Memo. at 9-11.) The Court of Appeal has specifically held that an injunction "need not specify in detail the processes whose use is prohibited." Components For Research, Inc. v. Isolation Prods., Inc., 241 Cal.App.2d 726, 731 (1966). Instead, it is

Foundstone has provided pages of detail describing the trade secrets at issue. Defendants' reliance on MAI Sys. Corp. v. Peak Computer, 991 F.2d 511 (9th Cir. 1993), is also misplaced. In MAI, the plaintiff cryptically referred only to "valuable trade secrets" in software, without any further explanation whatsoever.

[2] Defendants also assert that Foundstone has not complied with C.C.P. § 2019(d). This, too, is incorrect. Foundstone served a disclosure pursuant to Section 2019(d) on October 15, 2002.

sufficient to describe the technology and specifically limit the scope of an injunction to "method[s], technique[s], or processes . . . **developed by the plaintiff.**" Id. (Emphasis added.) For the additional reason that the trade secrets claimed here are all specifically limited to methods developed by Foundstone, the trade secrets are described in more-than-sufficient detail.

(b) The Trade Secrets Are Not Public Domain

Defendants also argue that Foundstone's trade secrets are "based on information available in the public domain." (Opp. at 10.) This assertion is utterly irrelevant. Foundstone does not dispute that its trade secrets rely, in part, on known principles. **Defendants, however, never claim that any of the publications or software they refer to in their papers discloses the actual methods Foundstone developed.**[3] They do not dispute that the massive investment Foundstone had to make to create its software. (McClure ¶ 4.) Indeed, Defendants admit that, to duplicate the functionality of Foundstone's software, it would be necessary for a competitor to make a similar investment of money and effort. (Glaser ¶ 31.)

(c) Reasonable Efforts To Maintain Secrecy

Defendants also assert that Foundstone did not take reasonable efforts to protect the secrecy of the software.

[3] The 152-page Exh. F to the Glaser Decl. is typically irrelevant to the trade secrets in this case. Defendants suggest that this document somehow discloses Foundstone's proprietary TCP and ICMP methods of operating system identification. The document, however, merely discusses general ICMP methods of network scanning, not the techniques developed by Foundstone.

(Opp. at 10.) Defendants do not dispute the comprehensive steps Foundstone has taken to protect its trade secrets. (McClure Decl. ¶¶ 13-20.) Instead, Defendants point only to Foundstone's alleged provision of evaluation copies of its software to reviewers, which would have theoretically allowed a reviewer to "read, write, or print" a single database. (Caso Decl. ¶ 15.) Defendants never claim that the data – an unintelligible mixed series of binary, decimal, and hexadecimal numbers – could be used or even understood by anyone. Defendants never state that any source code, or any means by which anyone could use or understand the allegedly compromised database was disclosed. Accordingly, there is no serious dispute that Foundstone took reasonable steps to protect its trade secrets.

2. **Defendants Have Misappropriated And Threatened To Misappropriate Foundstone's Trade Secrets**

In its moving papers, Foundstone submitted detailed evidence establishing that Defendants had misappropriated and threatened to misappropriate its trade secrets, citing to specific features of Defendants' software. (McClure Decl. ¶¶ 30-32.)[4] In their Opposition, Defendants do not dispute (indeed, they admit) that their software contains these features. (Caso Decl. Exh. C; Glaser Exh. E.) Moreover,

[4] Although Defendants assert objections to paragraphs 30-32 of the McClure Decl., they do not dispute the information presented. Moreover, the statements in paragraphs 30-32 of the McClure Decl. are confirmed by Defendants' own descriptions of their software. (Caso Decl. Exh. C; Glaser Decl. Exh. E.)

EXHIBIT 'F'

Darrell L. Olson, Bar No. 77,633
Michael K. Friedland, Bar No. 157,217
Douglas T. Hudson, Bar No. 210,385
KNOBBE, MARTENS, OLSON & BEAR, LLP
2040 Main Street
Fourteenth Floor
Irvine, California 92614
(949) 760-0404 (telephone)
(949) 760-9502 (facsimile)

Attorneys for Plaintiff FOUNDSTONE, INC.

SUPERIOR COURT OF THE STATE OF CALIFORNIA

COUNTY OF ORANGE, CENTRAL JUSTICE CENTER

FOUNDSTONE, INC., a Delaware corporation,

Plaintiff,

v.

NT Objectives, Inc., a California corporation; J.D. Glaser, an individual; Michael Morton, an individual; Eric Caso, an individual; Dan Kuykendall, an individual; and DOES 1 through 50, inclusive,

Defendants.

CASE NO. 02CC15350

ASSIGNED FOR ALL PURPOSES TO:
THE HON. DAVID H. BRICKNER
DEPT. C17

PLAINTIFF FOUNDSTONE'S SUPPLEMENTAL REPLY IN SUPPORT OF ORDER TO SHOW CAUSE RE: PRELIMINARY INJUNCTION

HEARING DATE: Oct. 25, 2002
TIME: 3:00 PM
DEPT: C17
TRIAL DATE: None

COMPLAINT FILED: Oct. 2, 2002

Plaintiff Foundstone, Inc. ("Foundstone") submits this Supplemental Reply in Support of the Order to Show Cause Re: Preliminary Injunction. The information addressed in this Supplemental Reply disproves a key assertion made by Defendants in their Opposition. The information was not discovered by Foundstone until after Foundstone filed its Reply on October 25, 2002.

In their Opposition, Defendants repeatedly argue that it is "impossible" for the Defendants to determine the trade secrets that are the subject of this action. (Opp. at 2. See also id. at 6-9.) The Defendants themselves submitted declarations, each asserting, in lock-step: "I have read the Declaration of Stuart McClure submitted in support of this OSC re: Preliminary Injunction. From review of that document I am unable to determine any specific method, algorithm or technique which is being referred to by Foundstone as a trade secret." (Caso Decl. ¶ 13; Glaser Decl. ¶ 26; Kuykendall Decl. ¶ 15; Morton Decl. ¶ 16.)

Information just uncovered by Foundstone demonstrates that these assertions are false. As set forth in Exh. A to the Hudson Decl. submitted herewith, Defendant Kuykendall maintains a personal site on the World Wide Web. **On his web site, Defendant Kuykendall published a paragraph-by-paragraph critique of the McClure Decl., specifically addressed each of the paragraphs of the McClure Decl. describing the trade secrets at issue in this action, and unambiguously admitted that he understood that Foundstone has proprietary technology in each described area.** The McClure Decl. described the trade secrets at issue in paragraphs 7-12. In his web site, Defendant Kuykendall responded to each of these paragraphs by admitting, inter alia, "they [Foundstone] have their proprietary implementation." (Hudson Decl. Ex. A p. 3, ¶¶ 7-12; emphasis added.)

For this additional reason, Foundstone requests that this Court preliminarily enjoin Defendants and maintain the status quo pending final adjudication of this matter.

Respectfully Submitted,

KNOBBE, MARTENS, OLSON & BEAR, LLP

Dated: 10/28/2002

By: ______________________

Darrell L. Olson
Michael K. Friedland
Douglas T. Hudson
Attorneys for Plaintiff
FOUNDSTONE, INC.

ICMP Usage in Scanning

Or

Understanding some of the ICMP Protocol's Hazards

Ofir Arkin

Founder

The Sys-Security Group

Sys-Security Group
BECAUSE SECURITY IS NOT TRIVIAL

http://www.sys-security.com
ofir@sys-security.com

Version 2.5

December 2000

Table of Contents

Introduction.. 7
1.1 Introduction to Version 1.0 7
1.2 Introduction to Version 2.0 7
1.3 Introduction to Version 2.5 8
1.4 CHANGES.. 8
1.4.1 Version 1.0 to Version 2.0.. 8
1.4.2 Version 2.0 to Version 2.01 9
1.4.3 Version 2.01 to Version 2.5 9
2.1 ICMP ECHO (Type 8) and ECHO Reply (Type 0) 10
2.2 ICMP Sweep (Ping Sweep).. 11
2.3 Broadcast ICMP.. 12
2.4 Non-ECHO ICMP.. 14
2.4.1 ICMP Time Stamp Request (Type 13) and Reply (Type 14).. 15
2.4.2 ICMP Information Request (Type 15) and Reply (Type 16) 16
2.4.3 ICMP Address Mask Request (Type 17) and Reply (Type 18) 19
2.5 Non-ECHO ICMP Sweeps 22
2.6 Non-ECHO ICMP Broadcasts 23
3.0 Advanced Host Detection using the ICMP Protocol (using ICMP Error Messages generated from the probed machines) 25
3.1 Sending IP Datagrams with bad IP headers fields – generating ICMP Parameter Problem error message back from probed machines 25
3.1.1 ACL Detection using IP Datagrams with bad IP headers fields 27
3.2 IP Datagrams with non-valid field values.. 29
3.2.1 The Protocol Field example.. 29
3.2.1.2 Using all combination of the IP protocol filed values.. 29
3.2.2 ACL Detection using the Protocol field 30
3.3 Host Detection using IP fragmentation to elicit Fragment Reassembly Time Exceeded ICMP error message. 31
3.3.1 ACL Detection using IP fragmentation 32
3.4 Host Detection using UDP Scans, or why we wait for the ICMP Port Unreachable 33
3.4.1 A Better Host Detection Using UDP Scan 34
3.5 Using Packets bigger than the PMTU of internal routers to elicit an ICMP Fragmentation Needed and Don't Fragment Bit was Set (configuration problem) 35
4.0 Inverse Mapping 36
4.1 Inverse Mapping Using ICMP (Echo & Echo Reply).. 36
4.2 Inverse Mapping Using Other Protocols.. 37
4.3 Patterns we might see 37
5.0 Using traceroute to Map a Network Topology 40
6.0 The usage of ICMP in Active Operating System Fingerprinting Process 43
Using Regular ICMP Query Messages 43
6.1.2 Using ICMP Information Requests 44
6.1.3 Identifying Operating Systems according to their replies for non-ECHO ICMP requests aimed at the broadcast address.. 44
6.2 The DF Bit Playground (Identifying Sun Solaris, HP-UX 10.30, 11.0x, and AIX 4.3.x based machines).. 45

6.2.1 Avoidance 52
6.3 The IP Time-to-Live Field Value with ICMP 53
6.3.1 IP TTL Field Value with ICMP Query Replies 53
6.3.2 IP TTL Field Value with ICMP ECHO Requests 55
6.3.3 Correlating the Information 56
6.4 Using Fragmented ICMP Address Mask Requests (Identifying Sun Solaris & HP-UX 11.0x machines) 57
Using Crafted ICMP Query Messages 59
Playing with the TOS Field 59
6.5 Precedence Bits Echoing (Fingerprinting Microsoft Windows 2000, ULTRIX, HPUX 11.0&10.30, OpenVMS and more) 61
6.5.1 Changed Pattern with other ICMP Query Message Types 67
6.6 TOSing OSs out of the Window / "TOS Echoing" (Fingerprinting Microsoft Windows 2000) 69
6.6.1 The use of the Type-of-Service field with the Internet Control Message Protocol 69
6.7 Using the TOS Byte's Unused Bit (Fingerprinting Microsoft Windows 2000, ULTRIX and more) 75
6.7.1 Changed Pattern with Replies for Different ICMP Query Types 77
6.8 Using the Unused (Identifying Sun Solaris & HP-UX 10.30 & 11.0x OS based machines) 78
6.9 DF Bit Echoing 80
6.9.1 DF Bit Echoing with the ICMP Echo request 80
6.9.2 DF Bit Echoing with the ICMP Address Mask request 81
6.9.3 DF Bit Echoing with the ICMP Timestamp request 81
6.9.4 Using all of the Information in order to identify maximum of operating systems 82
6.9.5 Why this would work (for the skeptical) 82
6.9.6 Combining all together 83
6.10 Using Code field values different than zero within ICMP ECHO requests 85
6.11 Using Code field values different than zero within ICMP Timestamp Request... 86
6.11.1 The non-answering Operating Systems 86
6.11.2 Operating Systems the Zero out the Code field value on Reply 87
Using the ICMP Error Messages 89
6.12 Operating system, which do not generate ICMP Protocol Unreachable Error Messages 89
6.13 ICMP Error Message Quenching 90
6.14 ICMP Error Message Quoting Size 90
6.15 LINUX ICMP Error Message Quoting Size Differences / The 20 Bytes from No Where 92
6.16 Foundry Networks Networking Devices Padded Bytes with ICMP Port Unreachable(s) / The 12 Bytes from No Where 94
6.17 ICMP Error Message Echoing Integrity (Tested with ICMP Port Unreachable).. 95
6.18 Novell Netware Echoing Integrity Bug with ICMP Fragment Reassembly Time Exceeded 100
6.19 The Precedence bits with ICMP Error Messages (Identifying LINUX) 101
6.20 TOS Bits (=field) Echoing with ICMP Error 104
6.21 DF Bit Echoing with ICMP Error Messages 105
Not that useful fingerprinting method(s) 112
6.22 Unusual Big ICMP Echo Request 112
7.0 Filtering ICMP on your Filtering Device to Prevent Scanning Using ICMP 114

7.1 Inbound 114
7.2 Outbound 114
7.3 Other Considerations 116
7.4 Other Problems – Why it is important to filter ICMP traffic in the Internal segmentation 117
7.5 The Firewall 118
8.0 Conclusion 120
9.0 Acknowledgment 121
9.1 Acknowledgment for version 1.0 121
9.1 Acknowledgment for version 2.0 121
9.2 Acknowledgment for version 2.5 121
Appendix A: The ICMP Protocol 122
A.1 ICMP Messages 123
A.1 ICMP Error Messages 125
A.1.1 ICMP Error Messages 126
A.1.1.1 Destination Unreachable (Type 3) 126
A.1.1.2 Source Quench (Type 4) 128
A.1.1.3 Redirect (Type 5) 129
A.1.1.4 Time Exceeded (Type 11) 130
A.1.1.5 Parameter Problem (Type 12) 130
Appendix B: ICMP "Fragmentation Needed but the Don't Fragment Bit was set" and the Path MTU Discovery Process 132
B.1 The PATH MTU Discovery Process 132
B.2 Host specification 132
B.3 Router Specification 133
B.4 The TCP MSS (Maximum Segment Size) Option and PATH MTU Discovery Process 134
Appendix C: Mapping Operating Systems for answering/discarding ICMP query message types 135
Appendix D: ICMP Query Message Types with Code field !=0 137
Appendix E: ICMP Query Message Types aimed at a Broadcast Address 139
Appendix F: Precedence Bits Echoing with ICMP Query Request & Reply 141
Appendix G: ICMP Query Message Types with TOS! = 0 142
Appendix H: Echoing the TOS Byte Unused bit 143
Appendix I: Using the Unused Bit 144
Appendix J: DF Bit Echoing 145
Appendix K: ICMP Error Message Echoing Integrity with ICMP Port Unreachable Error Message 146
Appendix L: Snort Basic Rule Base for ICMP Traffic 148

Figures List

Figure 1: ICMP ECHO Mechanism 10
Figure 2: ICMP ECHO Request & Reply message format 11
Figure 3: Broadcast ICMP 13
Figure 4: ICMP Time Stamp Request & Reply message format 15
Figure 5: ICMP Information Request and Reply 17
Figure 6: ICMP Address Mask Request & Reply message format 19
Figure 7: The IP Header 25
Figure 8: An Example: A TCP packet fragmented after only 8 bytes of TCP information 33
Figure 9: Using Packets bigger than the PMTU of internal routers to elicit an ICMP Fragmentation Needed and Don't Fragment Bit was Set 35
Figure 10: ICMP Time Exceeded message format 40
Figure 11: The Type of Service Byte 59
Figure 12: ICMP ECHO Request & Reply message format 86
Figure 13: The Type of Service Byte 101
Figure 14: Firewall ICMP Filtering Rules 117
Figure 15: Internal segmentation ICMP Filtering Example 118
Figure 16: ICMP Message Format 123
Figure 17: ICMP Error Message General Format 126
Figure 18: ICMP Fragmentation Needed but the Don't Fragment Bit was set Message Format 128
Figure 19: ICMP Redirect Message Format 129
Figure 20: ICMP Parameter Problem Message Format 131
Figure 21: ICMP Fragmentation Required with Link MTU 133

Table List

Table 1: Which Operating System would answer to an ICMP ECHO Request aimed at the Broadcast Address of the Network they reside on? 14
Table 2: Non-ECHO ICMP Query of different Operating Systems and Networking Devices 22
Table 3: Operating Systems, which would answer to requests, aimed at the Broadcast address 24
Table 4: Networking Devices, which would answer to requests, aimed at the Broadcast address 24
Table 5: IP TTL Field Values in replies from Various Operating Systems 53
Table 6: IP TTL Field Values in requests from Various Operating Systems 55
Table 7: Further dividing the groups of operating systems according to IP TTL field value in the ICMP ECHO Requests and in the ICMP ECHO Replies 56
Table 8: Precedence Field Values 60
Table 9: Type-of-Service Field Values 60
Table 10: ICMP Query Message Types with Precedence Bits ! = 0 68
Table 11: ICMP Query Message Types with TOS! = 0 75
Table 12: ICMP Query Message Types with the TOS Byte Unused Bit value ! = 0 78
Table 13: DF Bit Echoing 83
Table 14: ICMP Error Message Echoing Integrity 98
Table 15: Precedence Field Values 102
Table 16: ICMP message types 122
Table 17: ICMP Types & Codes 124
Table 18: Destination Unreachable Codes (Router) 127
Table 19: Redirect Codes 129
Table 20: Parameter Problem Codes 131

Diagram List

Diagram 1: The Inverse Mapping Idea 37
Diagram 2: A Decoy Scan Example 39
Diagram 3: Finger Printing Using ICMP Information Request Combines with ICMP Address Mask Request 44

Diagram 4: Finger Printing Using non-ECHO ICMP Query Types aimed at the Broadcast Address of an Attacked Network 45
Diagram 5: Finger Printing Using ICMP Address Mask Requests 59
Diagram 6: An example for a way to fingerprint Microsoft Windows 2000, Ultrix, HPUX 11.0 & 10.30, OpenVMS, Microsoft Windows ME, and Microsoft Windows 98/98SE based machines with ICMP Query messages with the Precedence Bits field !=0 68
Diagram 7: An example for a way to fingerprint Windows 2000, Ultrix, and Novell Netware based machines with ICMP Query messages with the TOS bits field !=0 74
Diagram 8: An example for a way to fingerprint operating systems using the unused bit in the TOS Byte echoing method 77
Diagram 9: An example of fingerprinting using the DF Bit Echoing technique 84
Diagram 10: Finger Printing Using ICMP Timestamp Request and Wrong Codes 87
Diagram 11: An Example of Finger Printing Using crafted ICMP Echo & Timestamp Request 89
Diagram 12: DF Bit Echoing with ICMP Error Messages 107

Introduction

1.1 Introduction to Version 1.0

The ICMP Protocol may seem harmless at first glance. Its goals and features were outlined in RFC 792 (and than later cleared in RFCs 1122, 1256, 1349, 1812), as a way to provide a means to send error messages. In terms of security, ICMP is one of the most controversial protocols in the TCP/IP protocol suite. The risks involved in implementing the ICMP protocol in a network, regarding scanning, are the subject of this research paper.

Scanning will usually be the major stage of an information gathering process a malicious computer attacker will lunch against a targeted network. With this stage the malicious computer attacker will try to determine what are the characteristics of the targeted network. He will use several techniques, such as host detection, service detection, network topology mapping, and operating system fingerprinting. The data collected will be used to identify those Hosts (if any) that are running a network service, which may have a known vulnerability. This vulnerability may allow the malicious computer attacker to execute a remote exploit in order to gain unauthorized access to those systems. This unauthorized access may become his focal point to the whole targeted network.

This research paper outlines the usage of the ICMP protocol in the scanning process. Step-by-Step we will uncover each of the malicious computer attacker techniques using the ICMP protocol. A few new scanning techniques will be unveiled in this research paper. I have reported some of them to several security mailing lists, including Bugtraq, in the past.

The chapters in this research paper are divided according to the various scanning techniques:

- Host Detection using the ICMP protocol is dealt in Chapter 2.
- Advanced Host Detection methods using the ICMP protocol are discussed in Chapter 3.
- Inverse Mapping using the ICMP protocol is discussed in Chapter 4.
- Network Mapping using the *traceroute* utility is discussed in Chapter 5.
- Chapter 6 discusses the usage of ICMP in the Active Operating System Fingerprinting process.
- Chapter 7 suggests a filtering policy to be used on filtering devices when dealing with the ICMP protocol.

I would like to take a stand in this controversial issue. ICMP protocol hazards are not widely known. I hope this research paper will change this fact.

1.2 Introduction to Version 2.0

Quite a large number of new OS fingerprinting methods using the ICMP protocol, which I have found are introduced with this revision. Among those methods two can be used in order to identify Microsoft Windows 2000 machines; one would allow us to distinguish between Microsoft Windows operating system machines and the rest of the world, and another would allow us to distinguish between SUN Solaris machines and the rest of the world[1]. I have also tried to be accurate as possible with data presented in this paper. Few tables have been added to the paper mapping the behavior of the various operating systems I have used. These tables describe the results I got from the various machines after querying them with the various tests introduced with this paper.

See section 1.3 for a full Changes list.

[1] See Section 6 for more information.

1.3 Introduction to Version 2.5

With this version of the research paper I am introducing a few new OS fingerprinting methods. Some are targeted in producing ICMP error messages from a target OS, enabling us to fingerprint an OS even if all ports of the OS in question are closed. I have also added a considerable amount of information about ICMP error message. At the end of the paper you will find the Basic snort rule base I have written.

1.4 CHANGES

1.4.1 Version 1.0 to Version 2.0

2.0 Host Detection Using the ICMP Protocol.

- 2.3 Broadcast ICMP
  - Added a table describing which operating systems would answer an ICMP ECHO request aimed at the Broadcast address of the network they reside on.
- 2.4 Non-ECHO ICMP
  - Added Information Request and Reply as a valid Host Detection method.
- 2.4.2 ICMP Information Request and Reply
  - The actual Information (added a section).
- 2.4.3 ICMP Address Mask Request and Reply
  - Added SUN Solaris and networking devices examples.
- 2.5 Non-Echo ICMP Sweep
  - Added a table summarizing which operating systems would answer those queries.
- 2.6 Non-ECHO ICMP Broadcasts
  - Added the fact that "Hosts running an operating system, which answers requests aimed at the IP broadcast address..."
  - Added two tables describing which operating systems would answer to which type of ICMP queries aimed at the broadcast address of the network they reside on?

3.0 Host Detection Using ICMP Error messages generated from the probed machines

- 3.1 IP datagrams with bad IP Header fields
  - Added more information on various other fields which can be used for this purpose.

6.0 The Usage of ICMP in the operating system Finger Printing Process

- 6.1 Using Wrong Codes within ICMP Datagrams
  - 6.1.1 Using ICMP Timestamp Requests with Codes different than 0
  - 6.1.2 Listing ICMP query message types sent to different operating systems with the Code field !=0 and the answers (is any) we got.
- 6.2 Using ICMP Address Mask Requests (Identifying Solaris Machines)
- 6.3 TOSing OSs out of the Window / Fingerprinting Microsoft Windows 2000
- 6.7 Using ICMP Address Mask Requests
- 6.8 Using ICMP Information Requests
- 6.9 Identifying operating systems according to their replies for non-ECHO ICMP requests aimed at the broadcast address.
- 6.10 IP TTL Field Value with ICMP
  - 6.10.1 IP TTL Field Value with ICMP ECHO Replies
  - 6.10.2 IP TTL Field Value with ICMP ECHO Requests

6.11 DF Bit
6.12 DF Bit Echoing
6.12.1 DF Bit Echoing with ICMP Echo requests
6.12.2 DF Bit Echoing with ICMP Address Mask requests
6.12.3 DF Bit Echoing with ICMP Timestamp requests
6.12.4 Using all of the Information in order to identify the maximum of operating systems.
6.12.5 Why this would work (for the skeptical)
6.13 What will not provide any gain compared to the effort and the detection ability?
6.13.1 Unusual big ICMP Echo messages

7.0 Filtering ICMP on your Filtering Device to Prevent Scanning Using ICMP

7.3 Other Considerations
More information was added.

Appendixes
Appendix C: Table - Mapping Operating Systems for answering/discarding ICMP query Message types.
Appendix D: Table - ICMP Query Message Types with Code Field !=0
Appendix E: Table - ICMP Query Message Types aimed at a Broadcast Address
Appendix F: Table - ICMP Query Message Types with TOS !=0
Appendix G: Table - DF Bit Echoing

1.4.2 Version 2.0 to Version 2.01

The Introduction was re-written

1.4.3 Version 2.01 to Version 2.5

To Section 4 "Inverse Mapping" more information and explanations were added.

Section 6 is now divided into four main subjects:

- Fingerprinting using regular ICMP Query requests
- Fingerprinting using crafted ICMP Query request
- Fingerprinting using ICMP Error Messages
- Not that useful fingerprinting methods

Multiple new fingerprinting methods based on ICMP Error Messages were introduced.

I have also introduced few Fingerprinting method based on ICMP Query messages: "Using the Unused (Identifying Sun Solaris & HP-UX 10.30 & 11.0x)", "Precedence Bits Echoing (Win2k, ULTRIX Identification)", "The TOS Byte Unused Bit Echoing (Identifying Win2k, ULTIX)".

"The DF Bit Playground" fingerprinting method was better explained and explored.

Appendix A now includes explanation for the various ICMP Error Messages.

2.0 Host Detection using the ICMP Protocol[2]

The Host Detection stage gives a malicious computer attacker crucial information by identifying the computers on the targeted network that are reachable from the Internet. This process belongs to the scanning stage, which is one of the first stages in the Information Gathering process. The information collected during this stage could later lead to an attempt to break in to one (or more) of the targeted network computers. This, if the information gathered would be sufficient for the malicious computer attacker.

2.1 ICMP ECHO (Type 8) and ECHO Reply (Type 0)

We can use an *ICMP ECHO* datagram to determine whether a target IP address is active or not, by simply sending an ICMP ECHO[3] (ICMP type 8) datagram to the targeted system and waiting to see if an *ICMP ECHO Reply* (ICMP type 0) is received. If an ICMP ECHO reply is received, it would indicate that the target is alive (few firewalls spoof ICMP ECHO replies from protected hosts); No response means the target is down or a filtering device is preventing the incoming ICMP ECHO datagram from getting inside the protected network or the filtering device prevents the initiated reply from reaching the Internet.

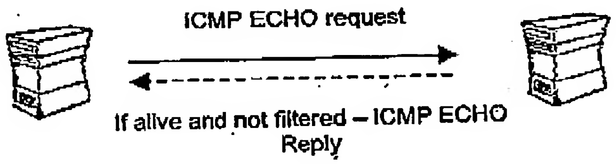


Figure 1: ICMP ECHO Mechanism

This mechanism is used by the Ping command to determine if a destination host is reachable.

In the next example two LINUX machines demonstrate the usage of Ping:

```
[root@stan /root]# ping 192.168.5.5
PING 192.168.5.5 (192.168.5.5) from 192.168.5.1 : 56(84) bytes of data.
64 bytes from 192.168.5.5: icmp_seq=0 ttl=255 time=4.4 ms
64 bytes from 192.168.5.5: icmp_seq=1 ttl=255 time=5.9 ms
64 bytes from 192.168.5.5: icmp_seq=2 ttl=255 time=5.8 ms

--- 192.168.5.5 ping statistics ---
3 packets transmitted, 3 packets received, 0% packet loss
round-trip min/avg/max = 4.4/5.3/5.9 ms
```

A Snort trace[4]:

```
01/26-13:16:25.746316 192.168.5.1 -> 192.168.5.5
```

[2] For more information about the ICMP Protocol please read "Appendix A: The ICMP Protocol".

[3] From a technical point of view: The sending side initializes the identifier (used to identify ECHO requests aimed at different destination hosts) and sequence number (if multiple ECHO requests are sent to the same destination host), adds some data (arbitrary) to the data field and sends the ICMP ECHO to the destination host. In the ICMP header the code equals zero. The recipient should *only change* the type to ECHO Reply and return the datagram to the sender.

[4] Snort, written by Martin Roesch, can be found at http://www.snort.org.

```
ICMP TTL:64 TOS:0x0 ID:6059
ID:5721    Seq:1  ECHO
89 D7 8E 38 27 63 0B 00 08 09 0A 0B 0C 0D 0E 0F   ...8'c..........
10 11 12 13 14 15 16 17 18 19 1A 1B 1C 1D 1E 1F   ................
20 21 22 23 24 25 26 27 28 29 2A 2B 2C 2D 2E 2F    !"#$%&'()*+,-./
30 31 32 33 34 35 36 37                           01234567

01/26-13:16:25.746638 192.168.5.5 -> 192.168.5.1
ICMP TTL:255 TOS:0x0 ID:6072
ID:5721    Seq:1  ECHO REPLY
89 D7 8E 38 27 63 0B 00 08 09 0A 0B 0C 0D 0E 0F   ...8'c..........
10 11 12 13 14 15 16 17 18 19 1A 1B 1C 1D 1E 1F   ................
20 21 22 23 24 25 26 27 28 29 2A 2B 2C 2D 2E 2F    !"#$%&'()*+,-./
30 31 32 33 34 35 36 37                           01234567
```

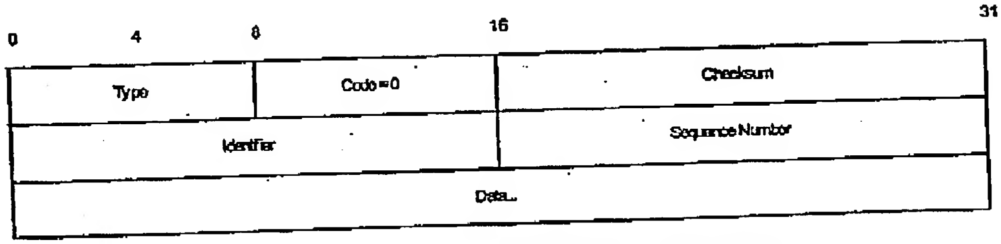


Figure 2: ICMP ECHO Request & Reply message format

Countermeasure: Block ICMP ECHO requests coming from the Internet towards your network at your border router and/or Firewall[5].

2.2 ICMP Sweep (Ping Sweep)

Querying multiple hosts using ICMP ECHO is referred to as *ICMP Sweep* (or *Ping Sweep*).

For a small to midsize network the Ping utility is an acceptable solution to this kind of host detection, but with large networks (such as Class A, or a full Class B) this kind of scan is fairly slow mainly because Ping waits for a reply (or a time out to be reached) from the probed host before proceeding to the next one.

fping[6] is a UNIX utility which sends parallel mass ECHO requests in a round robin fashion enabling it to be significantly faster than the usual Ping utility. It can also be fed with IP addresses with its accompanied tool *gping*. gping is used to generate a list of IP addresses which would be later fed into fping, directly or from a file, to perform the ICMP sweep. fping is also able to resolve hostnames of the probed machines if using the –d option.

Another UNIX tool that is able of doing an ICMP sweep in parallel, resolve the hostnames of the probed machines, save it to a file and a lot more is NMAP[7], written by Fyodor.

[5] It is better to filter unwanted traffic at your border router, reducing traffic rates for your firewall.

[6] ftp://ftp.tamu.edu/pub/Unix/src

[7] http://www.insecure.org

For the Microsoft Windows operating system a notable ICMP sweep tool is Pinger from Rhino9[8]. able of doing what fping and NMAP do regarding this kind of scan.

Trying to resolve the names of the probed machines may discover the malicious computer attacker's IP number used for the probing, using the log of the authoritative DNS server.

The next example demonstrates the usage of NMAP to perform an ICMP sweep[9] against 20 IP addresses. Our test lab contains two LINUX machines running Redhat Linux v6.1, Kernel 2.2.12 (Stan & Kenny) and one Windows NT WRKS SP4 (Cartman). As it can be seen all of the machines answered the probe:

```
[root@stan /root]# nmap -sP -PI 192.168.5.1-20

Starting nmap V. 2.3BETA13 by fyodor@insecure.org (
www.insecure.org/nmap/ )
Host stan.sys-security.com (192.168.5.1) appears to be up.
Host kenny.sys-security.com (192.168.5.5) appears to be up.
Host cartman.sys-security.com (192.168.5.15) appears to be up.
Nmap run completed -- 20 IP addresses (3 hosts up) scanned in 3 seconds
```

If we wish to avoid the automatic resolving done by NMAP we should use the –n option to eliminate it.

ICMP sweeps are easily detected by IDS (Intrusion Detection Systems) whether launched in the regular way, or if used in a parallel way.

Countermeasure: Block ICMP ECHO requests coming from the Internet towards your network at your border router and/or Firewall.

2.3 Broadcast ICMP

A simpler way to map a targeted network for alive hosts is by sending an ICMP ECHO request to the broadcast address or to the network address of the targeted network.

The request would be broadcasted to all hosts on the targeted network. The alive hosts will send an ICMP ECHO Reply to the prober's source IP address (additional conditions apply here).

The malicious computer attacker has to send only one IP packet to produce this behavior.

This technique of host detection is applicable only to some of the UNIX and UNIX-like hosts of the targeted network. Microsoft Windows based machines will not generate an answer (ICMP ECHO Reply) to an ICMP ECHO request aimed at the broadcast address or at the network address. They are configured not to answer those queries out-of-the box (This applies to all Microsoft Windows operating systems accept for Microsoft Windows NT 4.0 with service pack below SP4). This is not an abnormal behavior as RFC 1122[10] states that if we send an ICMP ECHO request to an IP Broadcast or IP Multicast addresses it *may* be silently discarded by a host.

[8] The Rhino9 group no longer exists. Their tools are available from a number of sites on the Internet.

[9] The –sP –PI options enable NMAP to perform only an ICMP Sweep. The default behavior when using the –sP option is different and includes the usage of TCP ACK host detection technique as well.

[10] RFC 1122: Requirements for Internet Hosts - Communication Layers, http://www.ietf.org/rfc/rfc1122.txt.

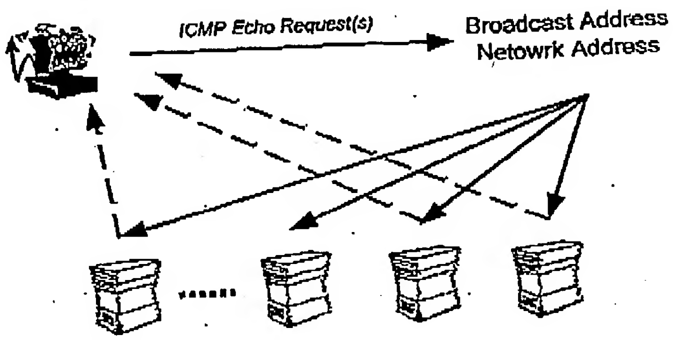


Figure 3: Broadcast ICMP

The next example demonstrates the behavior expected from hosts when sending an ICMP ECHO request to the broadcast address of a network. The two LINUX machines on our test lab answered the query while the Microsoft Windows NT 4.0 Workstation with SP6a machine silently ignored it.

```
[root@stan /root]# ping -b 192.168.5.255
WARNING: pinging broadcast address
PING 192.168.5.255 (192.168.5.255) from 192.168.5.1 : 56(84) bytes of
data.
64 bytes from 192.168.5.1: icmp_seq=0 ttl=255 time=4.1 ms
64 bytes from 192.168.5.5: icmp_seq=0 ttl=255 time=5.7 ms (DUP!)

--- 192.168.5.255 ping statistics ---
1 packets transmitted, 1 packets received, +1 duplicates, 0% packet
loss
round-trip min/avg/max = 4.1/4.9/5.7 ms
```

In the next example I have sent an ICMP ECHO request to the network address of the targeted network. The same behavior was produced. The LINUX machines answered the ICMP ECHO request while the Microsoft Windows NT 4.0 with SP6a machine ignored it.

```
[root@stan /root]# ping -b 192.168.5.0
WARNING: pinging broadcast address
PING 192.168.5.0 (192.168.5.0) from 192.168.5.1 : 56(84) bytes of data.
64 bytes from 192.168.5.1: icmp_seq=0 ttl=255 time=7.5 ms
64 bytes from 192.168.5.5: icmp_seq=0 ttl=255 time=9.1 ms (DUP!)

--- 192.168.5.0 ping statistics ---
1 packets transmitted, 1 packets received, +1 duplicates, 0% packet
loss
round-trip min/avg/max = 7.5/8.3/9.1 ms
```

Note: Broadcast ICMP may result in a *Denial-Of-Service* condition if a lot of machines response to the query at once.

A more accurate table that lists which operating systems would answer to an ICMP ECHO request aimed at their Network / Broadcast address is given below:

Operating System	Echo Request Broadcast
Debian GNU/ LINUX 2.2, Kernel 2.4 test 2	+
Redhat LINUX 6.2 Kernel 2.2.14	+
FreeBSD 4.0	-
FreeBSD 3.4	-
OpenBSD 2.7	-
OpenBSD 2.6	-
NetBSD	
Solaris 2.5.1	+
Solaris 2.6	+
Solaris 2.7	+
Solaris 2.8	+
HP-UX v10.20	+
AIX	
ULTRIX	
Windows 95	-
Windows 98	-
Windows 98 SE	-
Windows ME	-
Windows NT 4 WRKS SP 3	-
Windows NT 4 WRKS SP 6a	-
Windows NT 4 Server SP4	-
Windows 2000 Professional (and SP1)	-
Windows 2000 Server (and SP1)	-

Table 1: Which Operating Systems would answer to an ICMP ECHO Request aimed at the Broadcast Address of the Network they reside on?

Countermeasure: Block the IP directed broadcast on the border router.

2.4 Non-ECHO ICMP

ICMP ECHO is not the only ICMP query message type available with the ICMP protocol.

Non-ECHO ICMP messages are being used for more advanced ICMP scanning techniques (not only probing hosts, but network devices, such as a router, as well).

The group of ICMP query message types includes the following:

ECHO Request (Type 8), and Reply (Type 0)
Time Stamp Request (Type 13), and Reply (Type 14)
Information Request (Type 15), and Reply (Type 16)
Address Mask Request (Type 17), and Reply (Type 18)
Router Solicitation (Type 10), and Router Advertisement (Type 9)

2.4.1 ICMP Time Stamp Request (Type 13) and Reply (Type 14)

The *ICMP Time Stamp Request and Reply* allows a node to query another for the current time. This allows a sender to determine the amount of latency that a particular network is experiencing. The sender initializes the identifier (used to identify Timestamp requests aimed at different destination hosts) and sequence number (if multiple Timestamp requests are sent to the same destination host), sets the originate time stamp and sends it to the recipient.

The receiving host fills in the receive and transmit time stamps, change the type of the message to time stamp reply and returns it to the recipient. The time stamp is the number of milliseconds elapsed since midnight UT (GMT).

The originate time stamp is the time the sender last touched the message before sending it, the receive time stamp is the time the recipient first touched it on receipt, and the Transmit time stamp is the time the receiver last touched the message on sending it.

<table>
<tr><th>0</th><th>4</th><th>8</th><th>16</th><th>31</th></tr>
<tr><td colspan="2">Type</td><td>Code</td><td colspan="2">Checksum</td></tr>
<tr><td colspan="3">Identifier</td><td colspan="2">Sequence Number</td></tr>
<tr><td colspan="5">Originate timestamp</td></tr>
<tr><td colspan="5">Receive timestamp</td></tr>
<tr><td colspan="5">Transmit timestamp</td></tr>
</table>

Figure 4: ICMP Time Stamp Request & Reply message format

As RFC 1122 state, a *host may* implement Timestamp and Timestamp Reply. If they are implemented a host must follow this rules:

- Minimum variability delay in handling the Timestamp request.
- The receiving host *must* answer to every Timestamp request that he receives.
- An ICMP Timestamp Request to an IP Broadcast or IP Multicast address *may* be silently discarded.
- The IP source address in an ICMP Timestamp reply *must* be the same as the specific-destination address of the corresponding Timestamp request message.
- If a source-route option is received in a Timestamp request, the return route *must* be reserved and used as a Source Route option for the Timestamp Reply option.
- If a Record Route and/or Timestamp option is received in a Timestamp request, this option(s) *should* be updated to include the current host and included in the IP header of the Timestamp Reply message.

Receiving an ICMP Timestamp Reply would reveal an alive host (or a networking device) that has implemented the ICMP Timestamp messages.

In the next example I have sent an ICMP Time Stamp Request, using the icmpush[11] tool, to a Redhat 6.1 LINUX, Kernel 2.2.12 machine:

```
[root@stan /root]# icmpush -tstamp 192.168.5.5
kenny.sys-security.com -> 13:48:07
```

Snort Trace:

```
01/26-13:51:29.342647 192.168.5.1 -> 192.168.5.5
ICMP TTL:254 TOS:0x0 ID:13170
TIMESTAMP REQUEST
88 16 D8 D9 02 8B 63 3D 00 00 00 00 00 00 00 00   ......c=........

01/26-13:51:29.342885 192.168.5.5 -> 192.168.5.1
ICMP TTL:255 TOS:0x0 ID:6096
TIMESTAMP REPLY
88 16 D8 D9 02 8B 63 3D 02 88 50 18 02 88 50 18   ......c=..P...P.
2A DE 1C 00 A0 F9                                 *.....
```

When I have sent an ICMP Time Stamp Request to a Windows NT WRKS 4.0 SP4 machine, I got no reply. Again, this is not an abnormal behavior from the Microsoft Windows NT machine, just an implementation choice as RFC 1122 states.

Countermeasure: Block ICMP Time Stamp Requests coming from the Internet on the border Router and/or Firewall.

2.4.2 ICMP Information Request (Type 15) and Reply (Type 16)

The *ICMP Information Request/Reply* pair was intended to support self-configuring systems such as diskless workstations at boot time, to allow them to discover their network address.

The sender fills in the request with the Destination IP address in the IP Header set to zero (meaning this network). The request may be sent with both Source IP Address and Destination IP Address set to zero. The sender initializes the identifier and the sequence number, both used to match the replies with the requests, and sends out the request. The ICMP header code field is zero.

If the request was issued with a non-zero Source IP Address the reply would only contain the network address in the Source IP Address of the reply. If the request had both the Source IP Address and the Destination IP Address set to zero, the reply will contain the network address in both the source and destination fields of the IP header.

From the description above one can understand that the ICMP Information request and reply mechanism was intended to be used locally.

The RARP, BOOTP & DHCP protocols provide better mechanisms for hosts to discover its own IP address.

[11] Icmpush was written by Slayer of hispahack.http://hispahack.ccc.de/ .

0 4 8 16 31

Type	Code = 0	Checksum
Identifier		Sequence Number

Figure 5: ICMP Information Request & Reply message format

The Information Request & Reply mechanism is now obsolete as stated in RFC 1122, and RFC 1812[12]. A *router should not* originate or respond to these messages; A *host should not* implement these messages.

Demands on one hand and reality on the other.

RFC 792 specifies that the Destination IP address should be set to zero, this mean that hosts that do not reside on the same network cannot send these ICMP query type.

But what would happen if we would send an ICMP Information Request with the Destination IP address set to a specific IP address of a host out in the void?

The next example illustrates that some operating systems would answer these queries even if not issued from the same network. The ICMP Information Request queries we are sending are not really RFC compliant because of the difference in the Destination IP address.

Those operating systems that answer our queries work in contrast to the RFC guidelines as well. We would see in the next example why.

In the next example I have sent an ICMP Information Request, using the SING[13] tool, to an AIX machine:

```
[root@aik icmp]# ./sing -info host_address[14]
SINGing to host_address (ip_address): 8 data bytes
8 bytes from ip_address: icmp_seq=0 ttl=238 Info Reply
8 bytes from ip_address: icmp_seq=1 ttl=238 Info Reply
8 bytes from ip_address: icmp_seq=2 ttl=238 Info Reply
8 bytes from ip_address: icmp_seq=3 ttl=238 Info Reply

--- host_address sing statistics ---
5 packets transmitted, 4 packets received, 20% packet loss
```

The tcpdump trace:

```
19:56:37.943679 ppp0 > x.x.x.x > y.y.y.y: icmp: information request
                       4500 001c 3372 0000 ff01 18a7 xxxx xxxx
                       yyyy yyyy 0f00 bee3 321c 0000
19:56:38.461427 ppp0 < y.y.y.y > x.x.x.x: icmp: information reply
                       4500 001c 661b 0000 ee01 f6fd yyyy yyyy
```

[12] RFC 1812: Requirements for IP Version 4 Routers, http://www.ietf.org/rfc/rfc1812.txt . As the RFC states this mechanism is now obsolete - A *router should not* originate or respond to these messages; A *host should not* implement these messages.

[13] SING written by Alfredo Andres Omella, can be found at http://sourceforge.net/projects/sing.

[14] Since I have queried a production system for this test, with a permission of the owners, I do not wish to identify it.

```
xxxx xxxx 1000 bde3 321c 0000
```

Lets do a quick analysis of the trace.

The ICMP Information Request:

Value	Field	Additional Information
4	4-Bit Version	IP Version 4
5	4-Bit Header Length	4 x DWORD = 20 Bytes
00	8-Bit TOS	TOS=0
00 1c	16-Bit Total Length	
33 72	16-Bit Identification	
00 00	3-Bit Flags + 13-bit Fragment Offset	
ff	8-Bit TTL	TTL=255
01	8-Bit Protocol	1=ICMP
18 a7	16-Bit Header Checksum	
8b 5c d0 15	32-bit Source IP Address	139.92.208.21
xx xx xx xx	32-Bit Destination IP Address	
0f	8-Bit Type	Type=15
00	8-Bit Code	Code=0
be e3	16-Bit Checksum	
32 1c	16-Bit Identifier	
00 00	16-Bit Sequence Number	

The ICMP Information Reply:

Value	Field	Additional Information
4	4-Bit Version	IP Version 4
5	4-Bit Header Length	4 x DWORD = 20 Bytes
00	8-Bit TOS	TOS=0
00 1c	16-Bit Total Length	
66 1b	16-Bit Identification	
00 00	3-Bit Flags + 13-bit Fragment Offset	
ee	8-Bit TTL	TTL=238
01	8-Bit Protocol	1=ICMP
F6 fd	16-Bit Header Checksum	
xx xx xx xx	32-bit Source IP Address	
8b 5c d0 15	32-Bit Destination IP Address	139.92.208.21
10	8-Bit Type	Type=16
00	8-Bit Code	Code=0
bd e3	16-Bit Checksum	
32 1c	16-Bit Identifier	
00 00	16-Bit Sequence Number	

Instead of having the network address in the Source IP Address we are getting the IP address of the host.

Does the reply compliant with RFC 792 regarding this issue? Basically yes, because the RFC does not specify an accurate behavior.

The RFC states: "To form a information reply message, the source and destination addresses are simply reversed, the type code changes to 16, and the checksum recomputed".

This means that if the ICMP Information Request is coming from outside (Destination is not zero) of the network in question, the network address would not be revealed. But still a host could be revealed if he answers the request.

The request is not compliant with the RFC in my opinion because it does not fulfill its job – getting the network address.

Countermeasure: Block ICMP Information Requests coming from the Internet on the border Router and/or Firewall.

2.4.3 ICMP Address Mask Request (Type 17) and Reply (Type 18)

The *ICMP Address Mask Request* (and Reply) is intended for diskless systems to obtain its subnet mask in use on the local network at bootstrap time. Address Mask request is also used when a node wants to know the address mask of an interface. The reply (if any) contains the mask of that interface.

Once a host has obtained an IP address, it could than send an Address Mask request message to the broadcast address of the network they reside on (255.255.255.255). Any host on the network that has been configured to send address mask replies will fill in the subnet mask, change the type of the message to address mask reply and return it to the sender.

RFC 1122 states that the Address Mask request & reply query messages are entirely optional.

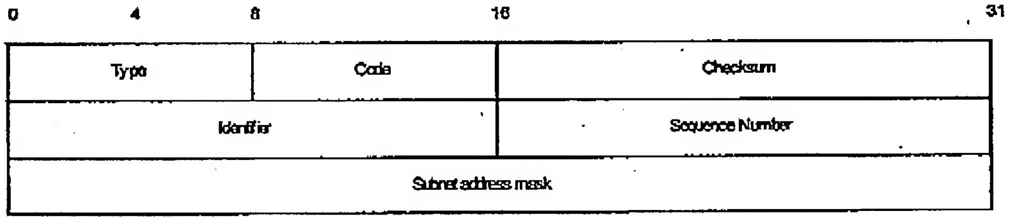


Figure 6: ICMP Address Mask Request & Reply message format

RFC 1122 also states that a system that has implemented ICMP Address Mask messages *must not* send an Address Mask Reply unless it is an authoritative agent for address masks.

Usually an Address Mask request would be answered by a gateway.

Receiving an Address Mask Reply from a host would reveal an alive host that is an authoritative agent for address masks. It will also allow a malicious computer attacker to gain knowledge about your network's configuration. This information can assist the malicious computer attacker in determining your internal network structure, as well as the routing scheme.

Please note that a Router *must* implement ICMP Address Mask messages. This will help identify routers along the path to the targeted network (it can also reveal internal routers if this kind of traffic is allowed to reach them).

If the Router is following RFC 1812 closely, it should not forward on an Address Mask Request to another network.

ICMP Address Mask Request aimed at a LINUX machine would not trigger an ICMP Address Mask Reply, nor a request aimed at a Microsoft Windows NT 4 Workstation SP 6a box.

In the next example I have sent an ICMP Address Mask Request to the broadcast address (192.168.5.255) of a class C network 192.168.5.0, spoofing the source IP to be 192.168.5.3:

```
[root@stan /root]# icmpush -vv -mask -sp 192.168.5.3 192.168.5.255
 -> ICMP total size = 12 bytes
 -> Outgoing interface = 192.168.5.1
 -> MTU = 1500 bytes
 -> Total packet size (ICMP + IP) = 32 bytes
ICMP Address Mask Request packet sent to 192.168.5.255 (192.168.5.255)

Receiving ICMP replies ...
-------------------------------------------------------
192.168.5.3 ...
   Type = Address Mask Request (0x11)
   Code = 0x0      Checksum = 0xBF87
     Id = 0x3B7        Seq# = 0x3CB0
-------------------------------------------------------
icmpush: Program finished OK
```

The snort trace:

```
-*> Snort! <*-
Version 1.5
By Martin Roesch (roesch@clark.net, www.clark.net/-roesch)
Kernel filter, protocol ALL, raw packet socket
Decoding Ethernet on interface eth0
02/15-13:47:37.179276 192.168.5.3 -> 192.168.5.255
ICMP TTL:254 TOS:0x0 ID:13170
ADDRESS REQUEST
B9 03 8E 49 00 00 00 00                           ...I....
```

No answer was received from the LINUX machines or from the Microsoft Windows NT Workstation 4 SP 6a machine on our test lab.

When I have tried to map which operating systems would answer (if at all) the ICMP Address Mask Requests, I have discovered that SUN Solaris is very cooperative with this kind of query[15]:

```
[root@aik icmp]# ./sing -mask -c 1 IP_Address[16]
SINGing to IP_Address (IP_Address): 12 data bytes
12 bytes from IP_Address: icmp_seq=0 ttl=241 mask=255.255.255.0

--- IP_Address sing statistics ---
1 packets transmitted, 1 packets received, 0% packet loss
[root@aik icmp]#
```

The Tcpdump trace:

[15] The –c 1 option enable SING to send only one ICMP datagram. The parameter can be changed to any desired value.

[16] The real IP Address and the Host address were replaced.

```
20:02:07.402229 ppp0 > x.x.x.x > y.y.y.y: icmp: address mask request
                          4500 0020 3372 0000 ff01 70a7 xxxx xxxx
                          yyyy yyyy 1100 afe3 3f1c 0000 0000 0000
20:02:07.831426 ppp0 < y.y.y.y > x.x.x.x: icmp: address mask is
0xffffff00 (DF)
                          4500 0020 3617 4000 f101 3c02 yyyy yyyy
                          xxxx xxxx 1200 afe2 3f1c 0000 ffff ff00
```

Our two last examples would be an ICMP Address Mask request aimed at a router (which must implement ICMP Address Mask Messages) and at a switch.

The following is an Address Mask Request sent to a Cisco Catalyst 5505 with OSS v4.5:

```
inferno:/tmp# sing -mask -c 1 10.13.58.240
SINGing to 10.13.58.240 (10.13.58.240): 12 data bytes
12 bytes from 10.13.58.240: icmp_seq=0 ttl=60 mask=255.255.255.0

--- 10.13.58.240 sing statistics ---
1 packets transmitted, 1 packets received, 0% packet loss
inferno:/tmp#

inferno:~# tcpdump -tnxv -s 1600 icmp
tcpdump: listening on xl0
10.13.58.199 > 10.13.58.240: icmp: address mask request (ttl 255, id
13170)
0000 :  4500 0020 3372 0000   FF01 FE99 0A0D 3AC7    E.. 3r........:.
0010 :  0A0D 3AF0 1100 6BF7   8308 0000 0000 0000    ..:...k.........

10.13.58.240 > 10.13.58.199: icmp: address mask is 0xffffff00 (ttl 60,
id 20187)
0000 :  4500 0020 4EDB 0000   3C01 A631 0A0D 3AF0    E.. N...<..1..:.
0010 :  0A0D 3AC7 1200 6BF6   8308 0000 FFFF FF00    ..:...k.........
0020 :  0000 0000 0000 0000   0000 0000 0000         ..............
^C
79 packets received by filter
0 packets dropped by kernel
inferno:~#
```

The last example is an ICMP Address Mask request sent to an Intel 8100 ISDN Router on our network:

```
[root@aik icmp]# ./sing -mask 10.0.0.254
SINGing to 10.0.0.254 (10.0.0.254): 12 data bytes
12 bytes from 10.0.0.254: icmp_seq=0 ttl=64 mask=255.255.255.0
12 bytes from 10.0.0.254: icmp_seq=1 ttl=64 mask=255.255.255.0
12 bytes from 10.0.0.254: icmp_seq=2 ttl=64 mask=255.255.255.0

--- 10.0.0.254 sing statistics ---
3 packets transmitted, 3 packets received, 0% packet loss
[root@aik icmp]#
```

The tcpdump trace:

```
[root@aik /root]# tcpdump -x icmp
Kernel filter, protocol ALL, datagram packet socket
tcpdump: listening on all devices
16:34:30.666687 eth0 > 10.0.0.105 > 10.0.0.254: icmp: address mask
request
                         4500 0020 3372 0000 ff01 7304 0a00 0069
                         0a00 00fe 1100 0afd e402 0000 0000 0000
16:34:30.667961 eth0 < 10.0.0.254 > 10.0.0.105: icmp: address mask is
0xffffff00
                         4500 0020 2cb7 0000 4001 38c0 0a00 00fe
                         0a00 0069 1200 0afc e402 0000 ffff ff00
                         0000 0000 0000 0000 0000 0000 0000
```

Countermeasure: Block ICMP Address Mask Requests coming from the Internet on the border Router and/or Firewall.

2.5 Non-ECHO ICMP Sweeps

We can query multiple hosts using a Non-ECHO ICMP query message type. This is referred as a Non-ECHO ICMP sweep.

Who would answer our query?

Hosts that answer to the following:

- Hosts that are in a listening state.
- Hosts running an operating system that implemented the Non-ECHO ICMP query message type that was sent.
- Hosts that are configured to reply to the Non-ECHO ICMP query message type (few conditions here as well, for example: RFC 1122 states that a system that implemented ICMP Address Mask messages *must not* send an Address Mask Reply unless it is an authoritative agent for address masks).

Given the conditions above, which host(s) would answer our queries?

Operating System	Info. Request	Time Stamp Request	Address Mask Request
Debian GNU/ LINUX 2.2, Kernel 2.4 test 2	-	+	-
Redhat LINUX 6.2 Kernel 2.2.14	-	+	-
FreeBSD 4.0	-	+	-
FreeBSD 3.4	-	+	-
OpenBSD	-	+	-
NetBSD			
Solaris 2.5.1	-	+	+
Solaris 2.6	-	+	+
Solaris 2.7	-	+	+
Solaris 2.8	-	+	+
HP-UX v10.20	+	+	-
AIX v4.x	+	+	-

Operating System	Info. Request	Time Stamp Request	Address Mask Request
ULTRIX 4.2 – 4.5	+	+	+
Windows 95	-	-	+
Windows 98	-	+	+
Windows 98 SE	-	+	+
Windows ME	-	+	-
Windows NT 4 WRKS SP 3	-	-	+
Windows NT 4 WRKS SP 6a		-	
Windows NT 4 Server SP 4	-	-	-
Windows 2000 Professional	-	+	-
Windows 2000 Server	-	+	-

Networking Devices	Info. Request	Time Stamp Request	Address Mask Request
Cisco Catalyst 5505 with OSS v4.5	+	+	+
Cisco Catalyst 2900XL with IOS 11.2	+	+	-
Cisco 3600 with IOS 11.2	+	+	-
Cisco 7200 with IOS 11.3	+	+	-
Intel Express 8100 ISDN Router	-	-	+

Table 2: non-ECHO ICMP Query of different Operating Systems and Networking Devices

Countermeasure: Block ICMP Information Requests, ICMP Address Mask Requests & ICMP Time Stamp Requests coming from the Internet on the border Router and/or Firewall.

2.6 Non-ECHO ICMP Broadcasts

We can send a Non-ECHO ICMP query message type to the broadcast address or to the network address of the targeted network.

The request would be broadcasted to all listening hosts on the targeted network.

Who would answer our query?

- Hosts that are in a listening state
- Hosts running an operating system that implemented the Non-ECHO ICMP query message type that was sent.
- Hosts that are configured to reply to the Non-ECHO ICMP query message type (few conditions here as well, for example: a host may discard Non-ECHO ICMP query message type requests targeted at the broadcast address. For example an ICMP Timestamp Request to an IP Broadcast or IP Multicast address *may* be silently discarded).

Given the conditions above, the answering hosts would almost always be UNIX and UNIX-like machines. SUN Solaris, HP-UX, and LINUX are the only operating systems, from the group of operating systems I have tested, that would answer to an ICMP Timestamp Request aimed at the broadcast address of a network. HP-UX would answer Information Requests aimed at the broadcast address of a network. Non-would answer to an ICMP Address Mask Request aimed at the broadcast address of a network.

Operating System	Info. Request Broadcast	Time Stamp Request Broadcast	Address Mask Request Broadcast
Debian GNU/ LINUX 2.2, Kemel 2.4 test 2	-	+	-
Redhat LINUX 6.2 Kemel 2.2.14	-	+	-
FreeBSD 4.0	-	-	-
FreeBSD 3.4			
OpenBSD 2.7	-	-	-
OpenBSD 2.6	-	-	-
NetBSD			
Solaris 2.5.1	+	+	-
Solaris 2.6	+	+	-
Solaris 2.7	+	+	-
Solaris 2.8	+	+	-
HP-UX v10.20	+	+	-
AIX 4.x			
ULTRIX 4.2 – 4.5			
Windows 95			
Windows 98	-	-	-
Windows 98 SE	-	-	-
Windows ME	-	-	-
Windows NT 4 WRKS SP 3	-	-	-
Windows NT 4 WRKS SP 6a			
Windows NT 4 Server SP 4	-	-	-
Windows 2000 Professional (& SP1)	-	-	-
Windows 2000 Server (& SP1)	-	-	-

Table 3: Operating Systems, which would answer to requests, aimed at the Broadcast address

Networking Devices	Info. Request Broadcast	Time Stamp Request Broadcast	Address Mask Request Broadcast
Cisco Catalyst 5505 with OSS v4.5	+	+	+
Cisco Catalyst 2900XL with IOS 11.2	+	-	-
Cisco 3600 with IOS 11.2	+	-	-
Cisco 7200 with IOS 11.3	+	-	-
Intel Express 8100 ISDN Router	-	-	-

Table 4: Networking Devices, which would answer to requests, aimed at the Broadcast address

Countermeasure: Block the IP directed broadcast on the border router. Block ICMP Information Requests, ICMP Address Mask Requests & ICMP Time Stamp Requests coming from the Internet on the border Router and/or Firewall.

3.0 Advanced Host Detection using the ICMP Protocol (using ICMP Error Messages generated from the probed machines)

The advanced host detection methods rely on the idea that we can use various methods in order to elicit an ICMP Error Message back from a probed machine and discover its existence. Some of the methods described here are:

- Mangling IP headers
    - Header Length Field
    - IP Options Field
- Using non-valid field values in the IP header
    - Using valid field values in the IP header
- Abusing Fragmentation
- The UDP Scan Host Detection method

With the first method we are using bad IP headers in the IP datagram that would generate an ICMP Parameter Problem error back from the probed machine to the source IP address of the probing datagram. The second method use non-valid field values in the IP header in order to force the probed machine to generate ICMP Destination Unreachable error message back to the malicious computer attacker. The third method discussed uses fragmentation to trigger an ICMP Fragment Reassembly Time Exceeded error message from the probed machine. The last method uses the UDP Scan method to elicit ICMP Port Unreachable error message back from a closed UDP port(s) on the probed host(s).

When using some of those methods we can determine if a filtering device is present and some can even discover the Access Control List a Filtering Device is forcing on the protected network.

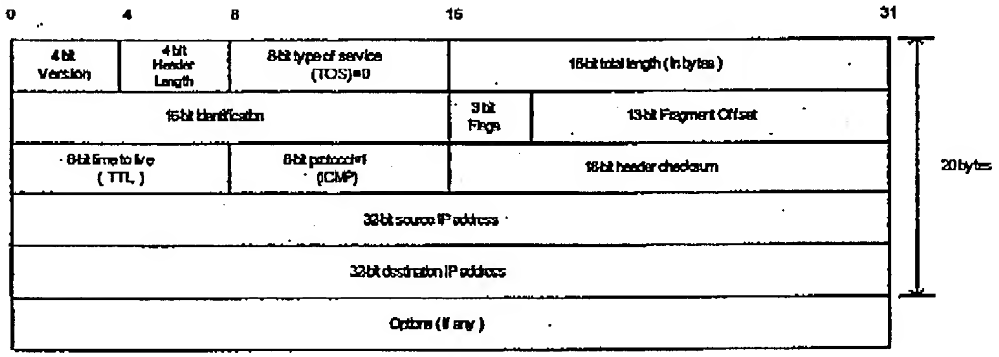


Figure 7: The IP Header

3.1 Sending IP Datagrams with bad IP headers fields – generating ICMP Parameter Problem error message back from probed machines

An ICMP Parameter Problem error message is sent when a router (*must* generate this message) or a host (*should* generate this message) process a datagram and finds a problem with the IP header parameters, which is not specifically covered by another ICMP error message. The ICMP Parameter Problem error message is only sent if the error caused the datagram to be discarded.

We have some variants with this type of Host Detection. We send an illegal forged datagram(s) with bad IP header field(s), that no specific ICMP error message is sent for this field(s). It will force a Host to send back an ICMP Parameter Problem Error message with either Code 0 or Code 2 (When code 0 is used, the pointer field will point to the exact byte in the original IP Header, which caused the problem. Code 2 is sent when the Header length or the total packet length values of the IP datagram do not appear to be accurate) to the source IP address of the bad IP datagram and reveal its existence. With this type of host detection it is not relevant what would be the protocol (TCP/UDP/ICMP) embedded inside the IP datagram. All we care about is the ICMP Error messages generated by the probed machine (if any).

This method is very powerful in detecting host(s) on the probed network with direct access from the Internet, since a host should generate this error message. Routers must generate the ICMP Parameter Problem error message as well, but not all of them check the correctness of some fields inside the IP header like a host does (processing of some fields is done on the host only).

According to RFC 1122 a host should check for validity of the following fields when processing a packet[17]:

- Version Number – if not 4 a host must silently discard the IP packet.
- Checksum – a host should verify the IP header checksum on every received datagram and silently discard every datagram that has a bad checksum.

A router should check for the validity of the following fields when processing a packet[18]:

- Checksum – a router must verify the IP checksum of any packet it received, and must discard messages containing invalid checksums.

The conditions outlined eliminate the usage of this method to a limited number of fields only.

It is possible to send an IP datagram with bad field(s) in the IP header, which will get routed without getting dropped in the way to the probed machine. It should be noted that different routers perform different checks regarding the IP header (different implementation and interpretation of RFC 1812). When a router, because of a bad IP header, drops an IP packet and sends an ICMP Parameter Problem error message, it is possible to identify the manufacture of the router, and to adjust the wrong IP header field correctly according to a field, which is not checked by the manufacture of that particular router.

A router may be more forgiving than a Host regarding the IP header. This may result from the fact that a router is a vehicle for delivering the IP datagram and a Host is the Destination and the place where more processing on the datagram is done.

The downside for this method is the detection. Intrusion Detection Systems *should* alert you about abnormalities in the attacked network traffic, since not every day you receive IP packets with bad IP Header field(s).

We can use this type of Host Detection to sweep through the entire IP range of an organization and get back results, which will map all the alive hosts on the probed network with direct access from the Internet.

Even if a firewall or another filtering device is protecting the probed network we can still try to send those forged packets to an IP addresses with ports that are likely to be opened. For

[17] RFC 1122 – Requirements for Internet Host, http://www.ietf.org/rfc/rfc1122.txt.

[18] RFC 1812 – Requirements for IPv4 Routers, http://www.ietf.org/rfc/rfc1812.txt.

example - TCP ports 21,25,80; UDP port 53; and even try to send an ICMP message presumably coming back from a Host/Router who generated it upon receiving data from the attacked network.

In my opinion Firewalls/Filtering Devices should check the validity of those fields used to elicit the ICMP Parameter Problem error message and disallow this kind of traffic.

An example is given here using the *ISIC* tool written by Mike Frantzen[19]. ISIC sends randomly generated packets to a target computer. Its primary uses are to stress test an IP stack, to find leaks in a firewall, and to test the implementation of Intrusion Detection Systems and firewalls. The user can specify how often the packets will be fragmented; have IP options, TCP options, an urgent pointer, etc.

In the next example I have sent 20 IP Packets from a LINUX machine to a Microsoft Windows NT WRKS 4 SP4 machine. The datagrams were not fragmented nor bad IP version numbers were sent. The only weird thing sent inside the IP headers was random IP Header length, which have produced ICMP Parameter Problem Code 2 error message as I anticipated.

```
[root@stan packetshaping]# ./isic -s 192.168.5.5 -d 192.168.5.15 -p 20
-F 0 -V 0 -I 100
Compiled against Libnet 1.0
Installing Signal Handlers.
Seeding with 2015
No Maximum traffic limiter
Bad IP Version   = 0%           Odd IP Header Length     = 100%
Frag'd Pcnt      = 0%

Wrote 20 packets in 0.03s @ 637.94 pkts/s
```

tcpdump trace:

```
12:11:05.843480 eth0 > kenny.sys-security.com > cartman.sys-
security.com: ip-proto-110 226 [tos 0xe6,ECT]  (ttl 110, id 119,
optlen=24[|ip])

12:11:05.843961 eth0 P cartman.sys-security.com > kenny.sys-
security.com: icmp: parameter problem - octet 21 Offending pkt:
kenny.sys-security.com > cartman.sys-security.com: ip-proto-110 226
[tos 0xe6,ECT]  (ttl 110, id 119, optlen=24[|ip]) (ttl 128, id 37776)
```

Other fields we can use inside the IP Header

In the last example we have used a bad Header Length field value to generate an ICMP Parameter Problem code 2-error message.

An ICMP Parameter Problem would almost always result from an incorrect usage of the *IP option* field as well.

3.1.1 ACL Detection using IP Datagrams with bad IP headers fields

If we probe the entire IP range of the targeted network with all combinations of protocols and ports, it would draw us the targeted network topology map, and will allow us to determine the access list (ACL) a Filtering Device (If present, and not blocking outgoing ICMP Parameter Problem Error messages) is forcing.

[19] http://expert.cc.purdue.edu/~frantzen/

This, if the filtering device does not check the validity of the mangled IP header fields, and allows the specified traffic.

3.1.1.1 How we determine the ACL (ICMP Protocol embedded inside)?

When the embedded protocol is ICMP, we send various ICMP message types encapsulated inside IP datagrams with bad IP header(s). If we receive a reply from a Destination IP address we have a host that is alive and an ACL, which allows this type of message of ICMP to get to the host who generated the ICMP error message (and the Parameter Problem ICMP error message, is allowed from the destination host to the Internet).

If we are not getting any reply than one of three possibilities:

- The Filtering Device disallows datagrams with the kind of bad field we are using.
- The Filtering Device is filtering the type of the ICMP message we are using.
- The Filtering Device blocks ICMP Parameter Problem error messages initiated from the protected network destined to the Internet.

3.1.1.2 How we determine the ACL (TCP or UDP Protocol embedded inside)?

We can probe for every combination of protocol and port values inside an IP packet with bad IP header(s). If we would receive an answer it would indicate that the protocol and port we used are allowed to the probed host from the Internet, and the ICMP Parameter Problem error message is allowed from the destination host in the protected network out to the Internet. It would also indicate that the filtering device used on the targeted network is not validating the correctness of the fields we have used in order to elicit the ICMP Parameter Problem error message.

If the embedded protocol were either TCP or UDP, a reply would not be generated if:

- The Filtering Device disallows packets with the kind of bad field we are using.
- The Filtering Device filters the Protocol used.
- The Filtering Device is filtering the specific port we are using for the probe.
- The Filtering Device blocks ICMP Parameter Problem error messages initiated from the protected network destined to the Internet. In our case, the filtering device may be blocking the specific host we are probing for outgoing ICMP Parameter Problem datagrams.

Note: If we are using the IP Header Length field in order to elicit ICMP Parameter Problem error message back from the probed host(s) than the host processing the datagram may not be able to access the Protocol Information embedded inside. The reason would be the faulty calculation that would be made – where the header ends and the data portion begins.

Countermeasure: Block outgoing ICMP Parameter Problem from the protected network to the Internet on the Firewall & on the border Router.

Check with the manufacture of your filtering device which fields it validates on the IP header when processing a datagram.

3.2 IP Datagrams with non-valid field values

This Host Detection method is based on different IP header fields within the crafted IP datagram that would have non-valid field values, which would trigger an ICMP Destination Unreachable Error message back from the probed machines.

Note that some hosts (AIX, HP-UX, Digital UNIX) may not send ICMP Protocol Unreachable messages.

3.2.1 The Protocol Field example

3.2.1.1 Using non-Valid (not used) IP protocol values

One such field within the IP header is the protocol field. If we will put a value, which does not represent a valid protocol number, the probed machine would elicit an ICMP Destination Unreachable – Protocol Unreachable error message back to the probed machine.

By sending this kind of crafted packets to all IP addresses within the IP address range of the probed network we can map the hosts that are directly connected to the Internet (assuming that no filtering device is present, or filtering the specific traffic).

3.2.1.1.1 Detecting if a Filtering Device is present

A packet sent with a protocol value, which does not represent a valid protocol number, should elicit an ICMP Destination Unreachable – Protocol Unreachable from the probed machine. Since this value is not used (and not valid) all hosts probed, unless filtered or are AIX, HP-UX, Digital UNIX machines, should send this reply. If a reply is not received we can assume that a filtering device prevents our packet from reaching our destination or from the reply to reach us back.

3.2.1.2 Using all combination of the IP protocol filed values

The difference with this variant is that we use all of the combinations available for the IP protocol field – since the IP protocol field has only 8 bits in length, there could be 256 combinations available.

NMAP 2.54 Beta 1 has integrated this variant and Fyodor have named it - IP Protocol scan. NMAP sends raw IP packets *without any further protocol header* (no payload) to each specified protocol on the target machine. If an ICMP Protocol Unreachable error message is received, the protocol is not in use. Otherwise it is assumed it is opened (or a filtering device is dropping our packets).

If our goal was Host Detection only, than using the NMAP implementation would be just fine. But if we wish to use this scan type for other purposes, such as ACL detection, than we would need the payload data as well (the embedded protocol's data).

We can determine if a filtering device is present quite easily using this scan method. If a large number of protocols (non valid values could be among those) seems to be "opened"/used (not receiving any reply – ICMP Protocol Unreachable) than we can assume a filtering device is blocking our probes (if using a packet with the protocol headers as well). If the filtering device is blocking the ICMP Protocol Unreachable error messages initiated from the protected network towards the Internet than nearly all of the 256 possible protocol values would be seemed "opened"/used.

With the current implementation with NMAP the 256 possible protocol values should be "opened" when a scan is performed against a machine inside a protected network, because a packet filter firewall (or other kind of firewall) *should* block the probe since it lacks information to validate the traffic against its rule base (information in the protocol headers such as ports for example).

In the next example I have used NMAP 2.54 Beta 1 in order to scan a Microsoft Windows 2000 Professional machine:

```
[root@catman /root]# nmap -vv -sO 192.168.1.1

Starting nmap V. 2.54BETA1 by fyodor@insecure.org (
www.insecure.org/nmap/ )
Host   (192.168.1.1) appears to be up ... good.
Initiating FIN,NULL, UDP, or Xmas stealth scan against  (192.168.1.1)
The UDP or stealth FIN/NULL/XMAS scan took 4 seconds to scan 254 ports.
Interesting protocols on  (192.168.1.1):
(The 250 protocols scanned but not shown below are in state: closed)
Protocol    State       Name
1           open        icmp
2           open        igmp
6           open        tcp
17          open        udp

Nmap run completed -- 1 IP address (1 host up) scanned in 4 seconds
```

A tcpdump trace of some of the communication exchanged:

```
17:44:45.651855 eth0 > localhost.localdomain > 192.168.1.1: ip-proto-50
0 (ttl 38, id 29363)
17:44:45.652169 eth0 < 192.168.1.1 > localhost.localdomain: icmp:
192.168.1.1 protocol 50 unreachable Offending pkt:
localhost.localdomain > 192.168.1.1: ip-proto-50 0 (ttl 38, id 29363)
(ttl 128, id 578)
17:44:45.652431 eth0 > localhost.localdomain > 192.168.1.1: ip-proto-
133 0 (ttl 38, id 18)
17:44:45.652538 eth0 > localhost.localdomain > 192.168.1.1: ip-proto-
253 0 (ttl 38, id 36169)
17:44:45.652626 eth0 > localhost.localdomain > 192.168.1.1: ip-proto-92
0 (ttl 38, id 26465)
17:44:45.652727 eth0 < 192.168.1.1 > localhost.localdomain: icmp:
192.168.1.1 protocol 133 unreachable Offending pkt:
localhost.localdomain > 192.168.1.1: ip-proto-133 0 (ttl 38, id 18)
(ttl 128, id 579)
17:44:45.652760 eth0 > localhost.localdomain > 192.168.1.1: ip-proto-
143 0 (ttl 38, id 14467)
17:44:45.652899 eth0 > localhost.localdomain > 192.168.1.1: ip-proto-30
0 (ttl 38, id 30441)
17:44:45.652932 eth0 < 192.168.1.1 > localhost.localdomain: icmp:
192.168.1.1 protocol 253 unreachable Offending pkt:
localhost.localdomain > 192.168.1.1: ip-proto-253 0 (ttl 38, id 36169)
(ttl 128, id 580)
```

3.2.2 ACL Detection using the Protocol field

First we need to determine if a filtering device is present using a non-valid (not used) protocol number probe. If a filtering device exists then no answer (ICMP Protocol Unreachable) will be received from the probed machine, assuming it is not AIX, HP-UX or Digital UNIX[20].

[20] You can determine this using OS finger printing methods.

If a certain protocol were not allowed through the filtering device we would not receive any ICMP error message from the probed machine. Probing for all combinations of protocols and ports against an IP range of a targeted network using non-valid and valid protocol values can determine the ACL a filtering device is forcing on the protected network, along with the topology map of a targeted network (hosts reachable from the Internet).

A reply would not be generated if:

- The Filtering Device filters the Protocol we are using
- The Filtering Device is filtering the specific port we are using for the probe.
- The Filtering Device blocks ICMP Destination Unreachable - Protocol Unreachable error messages initiated from the protected network destined to the Internet. In our case, the filtering device may be blocking the specific host we are probing for outgoing ICMP Destination Unreachable - Protocol Unreachable error messages.

Note: We can use this method for ACL detection but if the protocol we are using is not used on the target machine it should be blocked on the filtering device. Than, only opened TCP/UDP ports and allowed ICMP traffic could traverse the filtering device. If this kind of traffic is allowed we can have better ACL detection solutions then we outlined here.

Countermeasure: Block outgoing ICMP Protocol Unreachable error messages coming from the protected network to the Internet on your Firewall and/or Border Router. If you are using a firewall check that your firewall block protocols which are not supported (deny all stance).

3.3 Host Detection using IP fragmentation to elicit Fragment Reassembly Time Exceeded ICMP error message.

When a host receives a fragmented datagram with some of its pieces missing, and does not get the missing part(s) within a certain amount of time the host will discard the packet and generate an ICMP Fragment Reassembly Time Exceeded error message back to the sending host.

We can use this behavior as a Host Detection method, by sending fragmented datagrams with missing fragments to a probed host, and wait for an ICMP Fragment Reassembly Time Exceeded error message to be received from a live host(s), if any.

When we are using this method against all of the IP range of a probed network, we will discover the network topology of that targeted network.

In the next example I have sent a TCP fragment (with the MF bit set, using the –x option with hping2) to a Microsoft Windows ME machine:

```
[root@godfather bin]# hping2 -c 1 -x -y y.y.y.y
ppp0 default routing interface selected (according to /proc)
HPING y.y.y.y (ppp0 y.y.y.y): NO FLAGS are set, 40 headers + 0 data
bytes

--- y.y.y.y hping statistic ---
1 packets tramitted, 0 packets received, 100% packet loss
round-trip min/avg/max = 0.0/0.0/0.0 ms
[root@godfather bin]#
```

The tcpdump trace:

```
20:20:00.226064 ppp0 > x.x.x.x.1749 > y.y.y.y.0: .
1133572879:1133572879(0) win 512 (frag 31927:20@0+) (DF) (ttl 64)
                         4500 0028 7cb7 6000 4006 c8fd xxxx xxxx
                         d496 6607 06d5 0000 4390 f30f 0c13 6799
                         5000 0200 27a8 0000

20:21:00.033209 ppp0 < y.y.y.y > x.x.x.x: icmp: ip reassembly time
exceeded Offending pkt: [|tcp] (frag 31927:20@0+) (DF) (ttl 55) (ttl
119, id 12)
                         4500 0038 000c 0000 7701 6e9e yyyy yyyy
                         xxxx xxxx 0b01 b789 0000 0000 4500 0028
                         7cb7 6000 3706 d1fd xxxx xxxx yyyy yyyy
                         06d5 0000 4390 f30f
```

3.3.1 ACL Detection using IP fragmentation

This method can be used not only to map the entire topology map of the targeted network, but also to determine the ACL a firewall or a filtering device is forcing on the protected network.

Simply using all combinations of TCP and UDP with different ports, with the IP addresses from the IP range of the probed network will do it. When we receive a reply it means a host we queried is alive, the port we have used is opened on that host, and the ACL allows the protocol type and the port that was used to get to the probed machine (and the ICMP Fragment Reassembly Time Exceeded error message back from the probed machine to the Internet).

If we were not getting any reply back from the probed machine it can mean:

- The Filtering Device filters the Protocol used.
- The Filtering Device is filtering the specific port we are using for the probe.
- The Filtering Device blocks ICMP Fragment Reassembly Time Exceeded error messages initiated from the protected network destined to the Internet. In our case, the filtering device may be blocking the specific host we are probing for outgoing ICMP Parameter Problem datagrams.

3.3.1.1 An Example with UDP (Filtering Device Detection)

Since UDP is a stateless protocol it may be better suited for our needs here. The first datagram would be fragmented including enough UDP information in the first fragmented datagram that would be enough to verify the packet against a Firewall's Rule base. The second part of the datagram would not be sent. It would force any host that gets such a packet to send us back an ICMP Fragment Reassembly Time Exceeded error message when the time for reassembly exceeds.

If the port we were using were an open port, than the ICMP Fragment Reassembly Time Exceeded error message would be generated. If the port were closed then an ICMP Port Unreachable error message would be produced.

If a firewall is blocking our probed than *no reply* would be generated.

No reply would be an indication that traffic to the Host we probed is filtered.

3.3.1.2 An example with TCP

We can divide the first packet of the TCP handshake into two fragments. We would put enough TCP information in the first packet that would be enough to verify the packet against the Firewall's Rule base (this means the port numbers we are using are included in the packet). We will not send the second part of the packet, forcing any host that gets such a packet to send us back an ICMP Fragment Reassembly Time Exceeded error message when the time for reassembly exceeds. This would indicate the host is accessible by this kind of traffic, which is allowed using the port we have specified as the destination port[21].

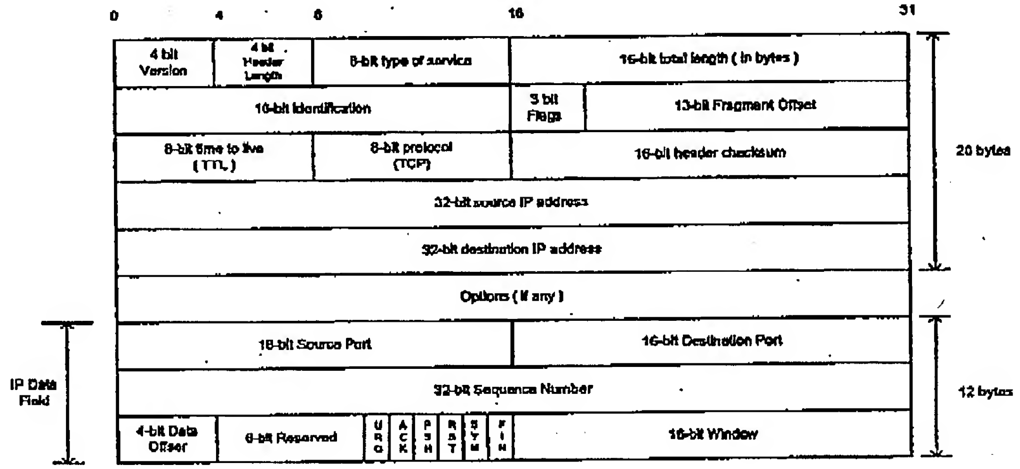


Figure 8: An Example: A TCP packet fragmented after only 12 bytes of TCP information

If the port we are using is open, than the ICMP error message would be generated. If the port is closed than a TCP RST packet should be sent back. If a filtering device were to block our probes than no reply would be generated. No reply would be an indication that traffic to the host we probed is filtered or the filtering device requires that the first TCP packet would not be fragmented (which is a legitimate requirement).

3.3.1.3 An Example with ICMP

We can do the same with encapsulating the ICMP protocol. When doing so the ICMP fragmented packets should sound the sirens when an Intrusion Detection system (If deployed) sees them. There is no reason to fragment an ICMP datagram.

If we think of sending fragmented ICMP through a bad filtering device product than we should at least include the first 4 bytes of the ICMP header with the IP datagram.

Countermeasure: Block outgoing ICMP Fragment Reassembly Time Exceeded Error messages.

3.4 Host Detection using UDP Scans, or why we wait for the ICMP Port Unreachable

How can we determine if a host is alive using a UDP probe? – We use the UDP scan method that uses ICMP Port Unreachable error message that may be generated from probed hosts as indicator of alive hosts. With this method we are sending a UDP datagram with 0 bytes of data to a UDP port on the attacked machine. If we have sent the datagram to a closed UDP port we will

[21] In a case were a firewall is validating that the first packet is not fragmented, we can fragment another one instead. But than this scanning method would not be any different from any other scanning method using TCP flags combinations.

receive an ICMP Port Unreachable error message. If the port is opened, we would not receive any reply.

When a filtering device is blocking UDP traffic aimed at the attacked machine, it would copycat the behavior pattern as with opened UDP ports.

If we probe a large number of UDP ports on the same host and we do not receive a reply from a large number of ports, it would look like that a large number of probed UDP ports are opened. While a filtering device is probably blocking the traffic and nearly all of the ports are closed.

How can we remedy this?
We can set a threshold number of non-answering UDP ports, when reached we will assume a filtering device is blocking our probes.

Fyodor has implemented a threshold with NMAP 2.3 BETA 13, so when doing a UDP scan and not receiving an answer from a certain number of ports, it would assume a filtering device is monitoring the traffic, rather than reporting those ports as opened.

3.4.1 A Better Host Detection Using UDP Scan

We will take the UDP scan method and tweak it a bit for our needs. We know that a closed UDP port will generate an ICMP Port Unreachable error message indicating the state of the port - closed UDP port. We will choose a UDP port that should be definitely closed (according to the IANA list of assigned ports ftp://ftp.isi.edu/in-notes/iana/assignments/port-numbers). For example we can use port 0 (but it would reveal our probe pretty easily).

Based on the fact that sending a UDP datagram to a closed port should elicit an ICMP Port Unreachable, we would send one datagram to the port we have chosen, than:

- If no filtering device is present we will receive an ICMP Port Unreachable error message, which will indicate that the Host is alive (or if this traffic is allowed by the filtering device).
- If no answer is given – a filtering device is covering that port.

In the next example I have used the HPING2[22] tool to send one UDP datagram to host 192.168.5.5 port 50, which was closed:

```
[root@stan /root]# hping2 -2 192.168.5.5 -p 50 -c 1
default routing not present
HPING 192.168.5.5 (eth0 192.168.5.5): udp mode set, 28 headers + 0 data
bytes
ICMP Port Unreachable from 192.168.5.5  (kenny.sys-security.com)

--- 192.168.5.5 hping statistic ---
1 packets tramitted, 0 packets received, 100% packet loss
round-trip min/avg/max = 0.0/0.0/0.0 ms


-*> Snort! <*-
Version 1.5
By Martin Roesch (roesch@clark.net, www.clark.net/~roesch)
Kernel filter, protocol ALL, raw packet socket
```

[22] HPING2 written by antirez, http://www.kyuzz.org/antirez/hping/ .

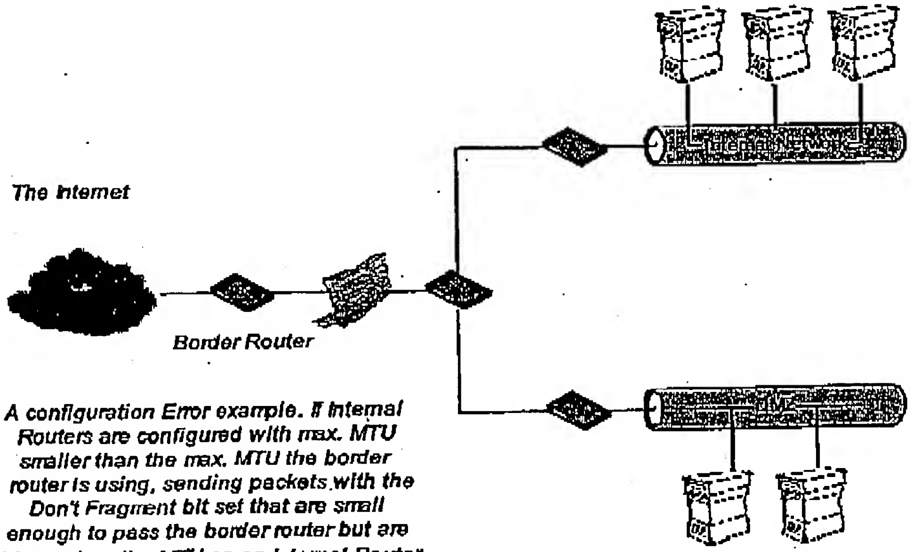
```
Decoding Ethernet on interface eth0
03/12-12:54:47.274096 192.168.5.1:2420 -> 192.168.5.5:50
UDP TTL:64 TOS:0x0 ID:57254
Len: 8

03/12-12:54:47.274360 192.168.5.5 -> 192.168.5.1
ICMP TTL:255 TOS:0xC0 ID:0
DESTINATION UNREACHABLE: PORT UNREACHABLE
00 00 00 00 45 00 00 1C DF A6 00 00 40 11 0F D4  ....E.......@...
C0 A8 05 01 C0 A8 05 05 09 74 00 32 00 08 6A E1  .........t.2..j.
```

We can use the port number we have chosen, or a list of UDP ports that are likely not being used, and query all the IP range of an attacked network. Getting a reply back would reveal a live host. No reply would mean a filtering device is covering those hosts UDP traffic, and probably other protocols and hosts as well.

3.5 Using Packets bigger than the PMTU of internal routers to elicit an ICMP Fragmentation Needed and Don't Fragment Bit was Set (configuration problem)

If internal routers have a PMTU that is smaller than the PMTU for a path going through the border router, those routers would elicit an ICMP "Fragmentation Needed and Don't Fragment Bit was Set" error message back to the initiating host if receiving a packet too big to process that has the Don't Fragment Bit set on the IP Header, discovering internal architecture of the router deployment of the attacked network.

This is in my opinion a configuration problem causing a security hazard.



A configuration Error example. If Internal Routers are configured with max. MTU smaller than the max. MTU the border router is using, sending packets with the Don't Fragment bit set that are small enough to pass the border router but are bigger than the MTU on an Internal Router would reveal those Router's existence.

Figure 9: Using Packets bigger than the PMTU of internal routers to elicit an ICMP Fragmentation Needed and Don't Fragment Bit was Set

4.0 Inverse Mapping

Inverse Mapping is a technique used to map internal networks or hosts that are protected by a filtering devices/firewall. Usually some of those systems are not reachable from the Internet. We use routers, which will give away internal architecture information of a network, even if the question they were asked does not make any sense, for this scanning type. We compile a list of IP's that list what is not there, and use it to conclude were things probably are.

We send a number of packets to different IP's we suspect are in the IP range of the network we are probing. When a router, either an exterior or interior, gets those packets for further processing, it looks at the IP address and makes decisions of routing based on it solely. When a router gets a packets with an IP which is not used in the IP space / network segment of the part of the probed network he serves, the router will elicit an ICMP Host Unreachable (Generated by a router if a route to the destination host on a directly connected network is not available – the destination host does not respond to ARP) or ICMP Time Exceeded error message(s) back to the originator of the datagram. If we do not get an answer about a certain IP we can assume this IP exist inside the probed network[23].

4.1 Inverse Mapping Using ICMP (Echo & Echo Reply)

Theoretically speaking, using any ICMP Query Message type or any ICMP Query Reply Message type in order to inverse map a network using a Router is possible.

With the next example I have sent an ICMP Echo Request to an IP that should be routed through a certain router (last hop before the host):

```
[root@cartman]# ./icmpush -vv -echo Target_IP[24]
 -> Outgoing interface = 192.168.1.5
 -> ICMP total size = 12 bytes
 -> Outgoing interface = 192.168.1.5
 -> MTU = 1500 bytes
 -> Total packet size (ICMP + IP) = 32 bytes
ICMP Echo Request packet sent to Target_IP (Target_IP)

Receiving ICMP replies ...
-----------------------------------------------------
Routers_IP  ...
          Type = Time Exceeded (0xB)
   Code = 0x0      Checksum = 0xF98F
     Id = 0x0           Seq# = 0x0
-----------------------------------------------------
./icmpush: Program finished OK


ICMP TTL:254 TOS:0x0 ID:13170
ID:12291   Seq:317  ECHO


02/13-09:16:31.724400 Routers_IP -> 192.168.1.5
ICMP TTL:57 TOS:0x0 ID:7410
```

[23] There is also a possibility that a filtering device is blocking our probes, or the replies.

[24] The real IP's of the targeted host and the Router were replaced because of legal problems that might arise when the ISP's personal that was used would understand it was one of their Routers used for this experiment.

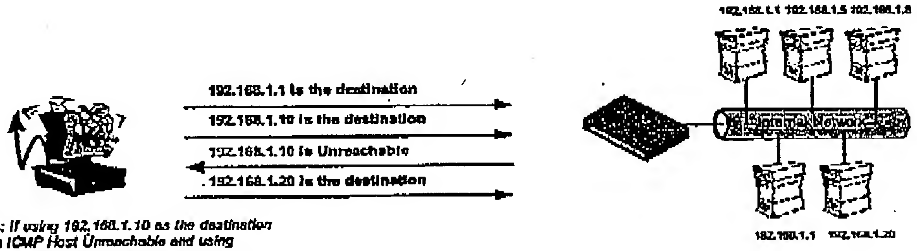
```
TTL EXCEEDED


00:13:12 prober> 192.168.2.5: icmp: echo reply
00:13:13 router> prober: icmp: host unreachable
```

Why Using ICMP Query Replies sometimes will be more beneficial than using ICMP Query Message types?

We have more chance of getting through filtering devices, that will allow replies of certain ICMP Query message types to get back to the issuing hosts. This might allow us to "penetrate" to deeper networks with our crafted reply.

4.2 Inverse Mapping Using Other Protocols

The technique of inverse mapping will work with other protocols as the stimulus. It will produce the same results since the destination Host (IP) will still be unreachable. The router one hop before the targeted host could not arp the host, and will issue an ICMP Host Unreachable regardless of the underlying protocol used.


192.168.1.1 192.168.1.5 192.168.1.8
192.168.1.1 is the destination
192.168.1.10 is the destination
192.168.1.10 is Unreachable
192.168.1.20 is the destination
Internal Network
192.168.1.1 192.168.1.20

Conclusion: If using 192.168.1.10 as the destination gave us an ICMP Host Unreachable and using 192.168.1.1 and 192.168.1.20 did not, then 192.168.1.1 and 192.168.1.20 are reachable and valid IPs within the targeted network address space

Diagram 1: The Inverse Mapping Idea

4.3 Patterns we might see

This type of scan will produce a numner of patterns. Not always, when we will see a router issuing a host unreachable it will be because some one ment to use the inverse mapping technique.

Lets look at our first example:

```
Router_IP > The_Same_IP : icmp: host Host_A unreachable
Router_IP > The_Same_IP : icmp: host Host_D unreachable
Router_IP > The_Same_IP : icmp: host Host_G unreachable
...
Router_IP > The_Same_IP : icmp: host Host_N unreachable
...
```

The same host is being used to scan an entire IP range of a targeted network. Some of the Hosts the malicious computer attacker tries to reach are not reachable. Still, the malicious computer attacker gets an idea about what is not reachable. Sometimes these results are the only indication that the malicious computer attacker will have about the presence of Hosts.

Lets look at the next example:

```
18:12:21.901256 Router_IP > 192.168.46.45: icmp: host x.x.x.12
unreachable
18:12:33.676136 Router_IP > 192.168.59.63: icmp: host x.x.x.12
unreachable
18:12:33.676218 Router_IP > 192.168.59.63: icmp: host x.x.x.12
unreachable
18:13:27.084221 Router_IP > 192.168.114.37: icmp: host x.x.x.12
unreachable
18:13:45.559706 Router_IP > 192.168.22.91: icmp: host x.x.x.12
unreachable
18:13:45.559856 Router_IP > 192.168.22.91: icmp: host x.x.x.12
unreachable
18:13:48.413514 Router_IP > 192.168.250.254: icmp: host x.x.x.12
unreachable
18:13:48.413681 Router_IP > 192.168.250.254: icmp: host x.x.x.12
unreachable
18:14:31.313495 Router_IP > 192.168.247.186: icmp: host x.x.x.12
unreachable
18:14:31.313624 Router_IP > 192.168.247.186: icmp: host x.x.x.12
unreachable
18:15:32.884187 Router_IP > 192.168.12.213: icmp: host x.x.x.12
unreachable
...
```

What we see here is different Hosts (changed to 192.168.x.x) failing to reach the x.x.x.12 IP. The Router is sending them all an ICMP Host Unreachable error message.

How come different Hosts (IPs) are seeking this host on such a short notice?

Probably what we are seeing is a decoy scan. A decoy scan is a type of scan, which involves multiple IPs, which are fed to the network-scanning tool as decoys. The real IP of the malicious computer attacker (or a machine he compromised) will be among those. For the defending side a question will be asked: What IP among all IPs, which are knocking on the door, is the IP the attacker was using?

With our example the IP is reported, to all seeking hosts, to be unreachable. The Router is trying to deliver the packet and fails with his ARP requests.

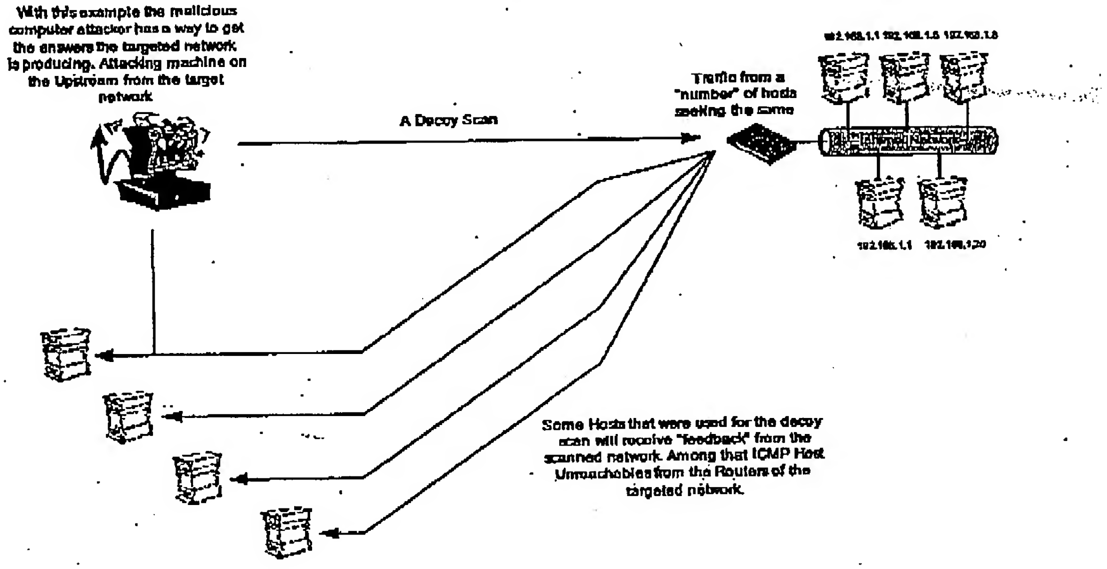

With this example the malicious computer attacker has a way to get the answers the targeted network is producing. Attacking machine on the Upstream from the target network
A Decoy Scan
Traffic from a "number" of hosts seeking the same
192.168.1.1 192.168.1.5 192.168.1.8
Internal Network
192.168.1.1 192.168.1.20
Some Hosts that were used for the decoy scan will receive "feedback" from the scanned network. Among that ICMP Host Unreachables from the Routers of the targeted network.

Diagram 2: A Decoy Scan Example

5.0 Using traceroute to Map a Network Topology

Traceroute is a Network debugging utility, which attempts to map all the hosts on a route to a certain destination host/machine.

The program sends UDP (by default) or ICMP ECHO Request[25] datagrams in sets of three, to a certain destination host. The first three datagram's to be sent have a Time-to-Live field value in the IP Header equals to one. The program lies on the fact that a router should decrement the TTL field value just before forwarding the datagram to another router/gateway.

If a router discovers that the Time-To-Live field value in an IP header of a datagram he process equals zero (or less) he would discard the datagram and generate an ICMP Time Exceeded Code 0 – transit TTL expired error message back to the originating host.

This is when a successful round is completed and another set of three datagrams is sent, this time with a Time-to-Live field value greater by one than the last set.

The originating host would know at which router the datagram expired since it receives this information with the ICMP Time Exceeded in Transit error message (Source IP address of the ICMP error message would be the IP address of the router/gateway; inside the IP header + 64 bits of original data of the datagram field we would have additional informaiton that would bound this ICMP error message to our issued traceroute command).

0	8	16 31
Type	Code	Checksum
Unused (zero)		
IP header + 64 bits of original data of the datagram		

Figure 10: ICMP Time Exceeded message format

Since we increment the TTL field starting from one for each successful round (again - a round is finished when the ICMP Time Exceeded in Transit error message is received) until we receive an ICMP Port Unreachable error message (or ICMP ECHO Reply if we are using the ICMP ECHO request datagrams) from the destined machine, we map every router/gateway/host along the path to our destination.

By default, when sending UDP packets we use a destination port which is probably not used by the destination host so the UDP datagram would not be processes and an ICMP Port Unreachable error message would be generated from the destined machine. The destination port would be incremented with each probe sent.

We get ICMP responses provided there is no prohibitive filtering or any packet loss.

[25] Microsoft Windows NT and Microsoft Windows 2000 are using the tracert command, which use ICMP ECHO Request datagrams as its default.

The output we see is a line showing the Time-To-Live, the address of the gateway, and the round trip time of each probe. If we do not get a response back within 5 seconds an "*" is printed, which represents no answer.

A regular traceroute example with ICMP would be[26]:

```
zuul:~>traceroute -I 10.0.0.10
traceroute to 10.0.0.10 (10.0.0.10), 30 hops max, 40 byte
packets
1 10.0.0.1 (10.0.0.1) 0.540 ms 0.394 ms 0.397 ms
2 10.0.0.2 (10.0.0.2) 2.455 ms 2.479 ms 2.512 ms
3 10.0.0.3 (10.0.0.3) 4.812 ms 4.780 ms 4.747 ms
4 10.0.0.4 (10.0.0.4) 5.010 ms 4.903 ms 4.980 ms
5 10.0.0.5 (10.0.0.5) 5.520 ms 5.809 ms 6.061 ms
6 10.0.0.6 (10.0.0.6) 9.584 ms 21.754 ms 20.530 ms
7 10.0.0.7 (10.0.0.7) 89.889 ms 79.719 ms 85.918 ms
8 10.0.0.8 (10.0.0.8) 92.605 ms 80.361 ms 94.336 ms
9 10.0.0.9 (10.0.0.9) 94.127 ms 81.764 ms 96.476 ms
10 10.0.0.10 (10.0.0.10) 96.012 ms 98.224 ms 99.312 ms
```

Lets assume that a network is protected by a firewall, which blocks all incoming traffic except for traffic aimed at the DNS Machine's UDP port 53. If we would perform a regular traceroute aimed for the DNS machine's IP address, our UDP datagrams would be sent with a destination port, which is probably not used on the targeted machine, and probably blocked by a Firewall or another filtering device. The traces would stop at the firewall at the entrance point to the probed network.

```
zuul:~>traceroute 10.0.0.10
traceroute to 10.0.0.10 (10.0.0.10), 30 hops max, 40 byte
packets
1 10.0.0.1 (10.0.0.1) 0.540 ms 0.394 ms 0.397 ms
2 10.0.0.2 (10.0.0.2) 2.455 ms 2.479 ms 2.512 ms
3 10.0.0.3 (10.0.0.3) 4.812 ms 4.780 ms 4.747 ms
4 10.0.0.4 (10.0.0.4) 5.010 ms 4.903 ms 4.980 ms
5 10.0.0.5 (10.0.0.5) 5.520 ms 5.809 ms 6.061 ms
6 10.0.0.6 (10.0.0.6) 9.584 ms 21.754 ms 20.530 ms
7 10.0.0.7 (10.0.0.7) 89.889 ms 79.719 ms 85.918 ms
8 10.0.0.8 (10.0.0.8) 92.605 ms 80.361 ms 94.336 ms
9 * * *
10 * * *
```

We need to set the port number to 53 in order to reach the DNS server. Since the traceroute program increases the port number every time it sends a UDP datagram, we need to calculate the port number to start with, so when a datagram would be processed by the Firewall[27] and would be examined, it would have the appropriate port and other information needed to fit with the Access Control List. If we use a simple equation we can calculate the starting port:

(Target port – (number of hops * number of probes)) -1

The number of hops (gateways) from our probing machine to the firewall is taken from our earlier traceroute. We use three probes for every query with the same TTL value, each one of them uses a different destination port number.

[26] All examples taken from "A Traceroute-Like Analysis of IP Packet Responses to Determine Gateway Access Control Lists" by David Goldsmith and Michael Shiffman. No real examples were provided because of legal issues.

[27] A firewall should not elicit any reply for any traffic destined directly for him.

```
zuul:~>traceroute -p28 10.0.0.10
traceroute to 10.0.0.10 (10.0.0.10), 30 hops max, 40 byte packets
1 10.0.0.1 (10.0.0.1) 0.501 ms 0.399 ms 0.395 ms
2 10.0.0.2 (10.0.0.2) 2.433 ms 2.940 ms 2.481 ms
3 10.0.0.3 (10.0.0.3) 4.790 ms 4.830 ms 4.885 ms
4 10.0.0.4 (10.0.0.4) 5.196 ms 5.127 ms 4.733 ms
5 10.0.0.5 (10.0.0.5) 5.650 ms 5.551 ms 6.165 ms
6 10.0.0.6 (10.0.0.6) 7.820 ms 20.554 ms 19.525 ms
7 10.0.0.7 (10.0.0.7) 88.552 ms 90.006 ms 93.447 ms
8 10.0.0.8 (10.0.0.8) 92.009 ms 94.855 ms 88.122 ms
9 10.0.0.9 (10.0.0.9) 101.163 ms * *
10 * * *
```

But with the regular traceroute program we now face another difficulty. After the datagram have passed the ACL of the Firewall (and we assume the firewall lets ICMP TTL Exceeded messages out) and listed the outer leg of the Firewall itself as the next hop, the next UDP datagram sent would be with a different port number - Than again it would be blocked by the firewall.

A modification to the traceroute program has been made by Michael Shiffman[28] in order to stop the port Incrementation. One side affect from sending traceroutes with a fixed port number, which is allowed on the firewalls ACL, is the final datagram, which normally would generate an ICMP Port Unreachable message now would not be generated since the UDP port would be in a listening state on the probed machine and would not provide an answer.

```
zuul:~>traceroute -S -p53 10.0.0.15
traceroute to 10.0.0.15 (10.0.0.15), 30 hops max, 40 byte
packets
1 10.0.0.1 (10.0.0.1) 0.516 ms 0.396 ms 0.390 ms
2 10.0.0.2 (10.0.0.2) 2.516 ms 2.476 ms 2.431 ms
3 10.0.0.3 (10.0.0.3) 5.060 ms 4.848 ms 4.721 ms
4 10.0.0.4 (10.0.0.4) 5.019 ms 4.694 ms 4.973 ms
5 10.0.0.5 (10.0.0.5) 6.097 ms 5.856 ms 6.002 ms
6 10.0.0.6 (10.0.0.6) 19.257 ms 9.002 ms 21.797 ms
7 10.0.0.7 (10.0.0.7) 84.753 ms * *
8 10.0.0.8 (10.0.0.8) 96.864 ms 98.006 ms 95.491 ms
9 10.0.0.9 (10.0.0.9) 94.300 ms * 96.549 ms
10 10.0.0.10 (10.0.0.10) 101.257 ms 107.164 ms 103.318 ms
11 10.0.0.11 (10.0.0.11) 102.847 ms 110.158 ms *
12 10.0.0.12 (10.0.0.12) 192.196 ms 185.265 ms *
13 10.0.0.13 (10.0.0.13) 168.151 ms 183.238 ms 183.458 ms
14 10.0.0.14 (10.0.0.14) 218.972 ms 209.388 ms 195.686 ms
15 10.0.0.15 (10.0.0.15) 236.102 ms 237.208 ms 230.185 ms
```

[28] http://www.packetfactory.net

6.0 The usage of ICMP in Active Operating System Fingerprinting Process

Finger Printing is the art of Operating System Detection.

A malicious computer attacker needs few pieces of information before lunching an attack. First, a target, a host detected using a host detection method. The next piece of information would be the services that are running on that host. This would be done with one of the Port Scanning methods. The last piece of information would be the operating system used by the host.

The information would allow the malicious computer attacker to identify if the targeted host is vulnerable to a certain exploit aimed at a certain service version running on a certain operating system.

In this section I have outlined the ICMP methods for this type of scan. Few methods are new and were discovered during this research.

Using Regular ICMP Query Messages

6.1 The "Who answer what?" approach

The question "Which operating system answer for what kind of ICMP Query messages?" help us identify certain groups of operating systems.

For example, LINUX and *BSD based operating systems with default configuration answer for ICMP Echo requests and for ICMP Timestamp Requests. Until Microsoft Windows 2000 family of operating systems has been released it was a unique combination for these two groups of operating systems. Since the Microsoft Windows 2000 operating system family mimics the same behavior (yes mimic), it is no longer feasible to make this particular distinction.

Microsoft might have been thinking that this way of behavior might hide Microsoft windows 2000 machines in the haze. As we will see with the examples given in this research paper they have much more to learn.

The thing is there is no clear distinction between one operating system to another based on this data. We can only group them together and try other methodologies in order to divide those groups a bit more[29].

Other data we might use is "Which operating system answer for queries aimed at the broadcast / network address of the network they reside on?"

For the complete mapping of the operating systems I have queried for this research please see "Appendix C: Mapping Operating Systems for answering/ discarding ICMP query message types", and "Appendix E: ICMP Query Message Types aimed at a Broadcast Address".

Two examples are given in this text for the usage of Operating System fingerprinting with the "Who answer what?" approach.

[29] Note: If the PMTU Discovery process using ICMP Echo requests is enables with HP-UX 10.30 & 11.0x operating systems than our simple query will trigger a "retaliation" from those machines, enabling us to identify them very easily. For more information on this issue see section 6.2

6.1.2 Using ICMP Information Requests

Because of the fact, that only few operating systems would reply to ICMP Information requests, we can group them together.

From the information given in table 2 in Section 2.5, we can conclude that HP-UX 10.20, AIX, ULTRIX & Open-VMS would be the only operating systems (among those I have tested) that would produce an ICMP Information reply for these queries.

We can further distinguish between those operating systems if we would send an ICMP Address Mask Request and wait for the reply from the systems in question. AIX and HP-UX operating systems would not answer the query, while the ULTRIX & Open-VMS would.

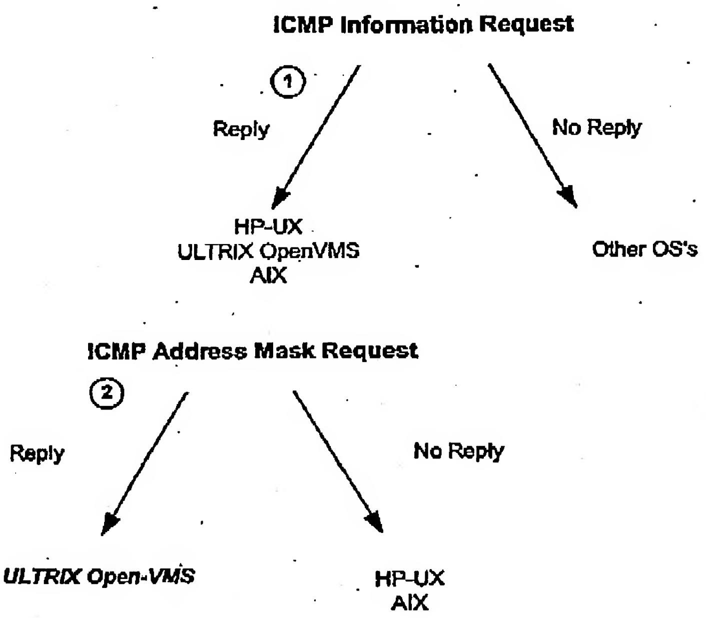


Diagram 3: Finger Printing Using ICMP Information Request Combined with ICMP Address Mask Request

6.1.3 Identifying Operating Systems according to their replies for non-ECHO ICMP requests aimed at the broadcast address

If IP directed broadcasts are not blocked, than we can identify the answering machines quite easily.

The first step is sending an ICMP Timestamp request aimed at the broadcast address of a targeted network. The operating systems who would answer would include SUN Solaris, HP-UX 10.20, and LINUX (Kernel version 2.2.x). We can further identify those operating systems by sending an ICMP Information request aimed at the broadcast address of the targeted network. HP-UX 10.20 would answer the query while SUN Solaris and LINUX would not. To distinguish between those two we would send an ICMP Address Mask request to the IPs that did not answer in the previous step. SUN Solaris would reply to the query while LINUX machines based on Kernel 2.2.x would not.

For complete information see Section 2.6.

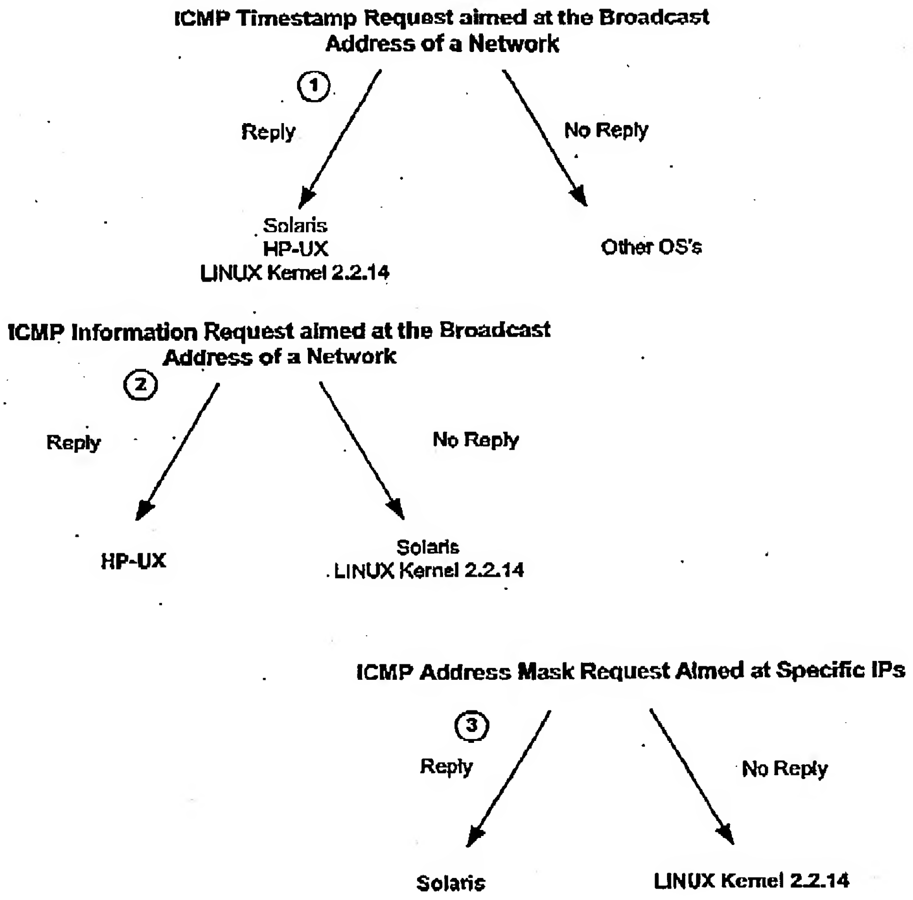

ICMP Timestamp Request aimed at the Broadcast Address of a Network
1
Reply
No Reply
Solaris
HP-UX
LINUX Kernel 2.2.14
Other OS's
ICMP Information Request aimed at the Broadcast Address of a Network
2
Reply
No Reply
HP-UX
Solaris
LINUX Kernel 2.2.14
ICMP Address Mask Request Aimed at Specific IPs
3
Reply
No Reply
Solaris
LINUX Kernel 2.2.14

Diagram 4: Finger Printing Using non-ECHO ICMP Query Types aimed at the Broadcast Address of an Attacked Network

6.2 The DF Bit Playground (Identifying Sun Solaris, HP-UX 10.30, 11.0x, and AIX 4.3.x based machines)

RFC 791 defines a three bits field used for various control flags in the IP Header.

Bit 0 is the reserved flag, and must be zero.

Bit 1, is called the Don't Fragment flag, and can have two values. A value of zero (not set) is equivalent to May Fragment, and a value of one is equivalent to Don't Fragment. If this flag is set than the fragmentation of this packet at the IP level is not permitted, otherwise it is.

Bit 2, is called the More Fragments bit. It can have two values. A value of zero is equivalent to (this is the) Last Fragment, and a value of 1 is equivalent to More Fragments (are coming).

The next field in the IP header is the Fragment Offset field, which identifies the fragment location relative to the beginning of the original un-fragmented datagram (RFC 791, bottom of page 23).

A close examination of the ICMP Query replies would reveal that some operating systems would set the DF bit with their replies.

The tcpdump trace below illustrates the reply a Sun Solaris 2.7 box produced for an ICMP Echo Request:

```
17:10:19.538020 if 4  > y.y.y.y > x.x.x.x : icmp: echo request (ttl
255, id 13170)
                                4500 0024 3372 0000 ff01 9602 yyyy yyyy
                                xxxx xxxx 0800 54a4 8d04 0000 cbe7 bc39
                                8635 0800
17:10:19.905254 if 4  < x.x.x.x > y.y.y.y : icmp: echo reply (DF) (ttl
233, id 24941)
                                4500 0024 616d 4000 e901 3e07 xxxx xxxx
                                yyyy yyyy 0000 5ca4 8d04 0000 cbe7 bc39
                                8635 0800
```

In the recent SING CVS (12 September 2000), written by Alfredo Andres Omella, which is available from http://sourceforge.net/projects/sing, the option for detecting if the DF bit is set with an ICMP Query reply was added, after being request by me. The following is the same ICMP Echo request & reply, this time it is presented by SING:

```
[root@godfather bin]# ./sing -echo Host_Address
SINGing to www.openbsd.org (IP_Address): 16 data bytes
16 bytes from IP_Address: icmp_seq=0 DF! ttl=233 TOS=0 time=367.314 ms
16 bytes from IP_Address: icmp_seq=1 DF! ttl=233 TOS=0 time=320.020 ms
16 bytes from IP_Address: icmp_seq=2 DF! ttl=233 TOS=0 time=370.037 ms
16 bytes from IP_Address: icmp_seq=3 DF! ttl=233 TOS=0 time=330.025 ms

--- Host_Address sing statistics ---
4 packets transmitted, 4 packets received, 0% packet loss
round-trip min/avg/max = 320.020/346.849/370.037 ms
```

Since www.openbsd.org uses a Sun Solaris operating system, this matches our findings.

ICMP Query replies for an operating system maintains the same behavioral patterns. Either they set the DF bit on all ICMP query reply types or they do not.

The DF bit would be set by default with ICMP Query replies with Sun Solaris. With HP-UX 10.30, & 11.0x, and with AIX 4.3.x setting the DF Bit will vary from one queried host to another (explanation coming). It may be set with the first ICMP Query reply onwards, or after a number of ICMP Query replies. This detail will help us to distinguish between Sun Solaris, HP-UX 10.30 & 11.0x, and AIX 4.3.x operating systems.

Why HP-UX 10.30 / 11.0 & AIX 4.3.x operating systems act this way?
HP claims to have a proprietary method in order to determine the PMTU with HP-UX v10.30, and HP-UX v11.0x using ICMP Echo requests. AIX 4.3.x do exactly the same.

The next trace will help understanding the process taken by HPUX 10.30 & 11.0x and AIX 4.3.x.

Here I have sent an ICMP Echo request to an HP-UX 11.0 machine:

```
00:27:56.884147 ppp0 > y.y.y.y > x.x.x.x: icmp: echo request (ttl 255,
id 13170)
                        4500 0024 3372 0000 ff01 7c51 yyyy yyyy
                        xxxx xxxx.0800 5238 6d04 0000 dce5 c339
                        8b7d 0d00
00:27:57.165620 ppp0 < x.x.x.x > y.y.y.y : icmp: echo reply (ttl 236,
id 41986)
                        4500 0024 a402 0000 ec01 1ec1 xxxx xxxx
                        yyyy yyyy 0000 5a38 6d04 0000 dce5 c339
                        8b7d 0d00
```

The first pair of ICMP Echo request and ICMP Echo reply was pretty usual. My LINUX machine sent an ICMP Echo request and received an ICMP Echo reply from the probed machine. One notable detail – the DF bit is not set in the ICMP Echo reply.

Than something that was not expectable has happened:

```
00:27:57.435620 ppp0 < x.x.x.x > y.y.y.y : icmp: echo request (DF) (ttl
236, id 41985)
                        4500 05dc a401 4000 ec01 d909 xxxx xxxx
                        yyyy yyyy 0800 7e52 9abc def0 0000 0000
                        0000 0000 0000 0000 0000 0000 0000 0000
                        0000 0000 0000 0000 0000 0000 0000 0000
                        0000 0000 0000 0000 0000 0000 0000 0000
                        0000 0000 0000 0000 0000 0000 0000 0000
                        0000 0000 0000 0000 0000 0000 0000 0000
                        0000 0000 0000 0000 0000 0000 0000 0000
                        0000 0000 0000 0000 0000 0000 0000 0000
                        0000 0000 0000 0000 0000 0000 0000 0000
                        0000 0000 0000 0000 0000 0000 0000 0000
                        0000 0000 0000 0000 0000 0000 0000 0000
                        0000 0000 0000 0000 0000 0000 0000 0000
                        0000 0000 0000 0000 0000 0000 0000 0000
                        0000 0000 0000 0000 0000 0000 0000 0000
                        0000 0000 0000 0000 0000 0000 0000 0000
                        0000 0000 0000 0000 0000 0000 0000 0000
                        0000 0000 0000 0000 0000 0000 0000 0000
                        0000 0000 0000 0000 0000 0000 0000 0000
                        0000 0000 0000 0000 0000 0000 0000 0000
                        0000 0000 0000 0000 0000 0000 0000 0000
                        0000 0000 0000 0000 0000 0000 0000 0000
                        0000 0000 0000 0000 0000 0000 0000 0000
                        0000 0000 0000 0000 0000 0000 0000 0000
                        0000 0000 0000 0000 0000 0000 0000 0000
                        0000 0000 0000 0000 0000 0000 0000 0000
                        0000 0000 0000 0000 0000 0000 0000 0000
```

```
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
```

```
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000
```

```
00:27:57.435672 ppp0 > y.y.y.y > x.x.x.x: icmp: echo reply (ttl 255, id
53)
                        4500 05dc 0035 0000 ff01 a9d6 yyyy yyyy
                        xxxx xxxx 0000 8652 9abc def0 0000 0000
                        0000 0000 0000 0000 0000 0000 0000 0000
                        0000 0000 0000 0000 0000 0000 0000 0000
                        0000 0000 0000 0000 0000 0000 0000 0000
                        0000 0000 0000 0000 0000 0000 0000 0000
                        0000 0000 0000 0000 0000 0000 0000 0000
                        0000 0000 0000 0000 0000 0000 0000 0000
                        0000 0000 0000 0000 0000 0000 0000 0000
                        0000 0000 0000 0000 0000 0000 0000 0000
                        0000 0000 0000 0000 0000 0000 0000 0000
                        0000 0000 0000 0000 0000 0000 0000 0000
                        0000 0000 0000 0000 0000 0000 0000 0000
                        0000 0000 0000 0000 0000 0000 0000 0000
                        0000 0000 0000 0000 0000 0000 0000 0000
                        0000 0000 0000 0000 0000 0000 0000 0000
                        0000 0000 0000 0000 0000 0000 0000 0000
                        0000 0000 0000 0000 0000 0000 0000 0000
                        0000 0000 0000 0000 0000 0000 0000 0000
                        0000 0000 0000 0000 0000 0000 0000 0000
                        0000 0000 0000 0000 0000 0000 0000 0000
                        0000 0000 0000 0000 0000 0000 0000 0000
                        0000 0000 0000 0000 0000 0000 0000 0000
                        0000 0000 0000 0000 0000 0000 0000 0000
                        0000 0000 0000 0000 0000 0000 0000 0000
                        0000 0000 0000 0000 0000 0000 0000 0000
                        0000 0000 0000 0000 0000 0000 0000 0000
                        0000 0000 0000 0000 0000 0000 0000 0000
                        0000 0000 0000 0000 0000 0000 0000 0000
                        0000 0000 0000 0000 0000 0000 0000 0000
                        0000 0000 0000 0000 0000 0000 0000 0000
                        0000 0000 0000 0000 0000 0000 0000 0000
                        0000 0000 0000 0000 0000 0000 0000 0000
                        0000 0000 0000 0000 0000 0000 0000 0000
                        0000 0000 0000 0000 0000 0000 0000 0000
                        0000 0000 0000 0000 0000 0000 0000 0000
                        0000 0000 0000 0000 0000 0000 0000 0000
                        0000 0000 0000 0000 0000 0000 0000 0000
                        0000 0000 0000 0000 0000 0000 0000 0000
                        0000 0000 0000 0000 0000 0000 0000 0000
                        0000 0000 0000 0000 0000 0000 0000 0000
                        0000 0000 0000 0000 0000 0000 0000 0000
                        0000 0000 0000 0000 0000 0000 0000 0000
                        0000 0000 0000 0000 0000 0000 0000 0000
```

```
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000
```

The HPUX 11.x machine I have queried pinged me back! The ICMP Echo request size was 1500 bytes. It was the maximum transfer unit my Internet Connection was allowed to process. The request was sent with the DF bit set. Any router along the way, trying to fragment the request because the MTU of the destined network was smaller than the datagram's size would fail and send an ICMP Error message back stating a fragmentation was required but the don't fragment bit was set. It would allow the sending machine to send a smaller sized datagram according to its PMTU discovery process/algorithm with ICMP. If for this ICMP Echo request an ICMP Echo reply would be received, than the PMTU is discovered.

```
00:27:57.885662 ppp0 > y.y.y.y > x.x.x.x : icmp: echo request (ttl 255,
id 13170)
                         4500 0024 3372 0000 ff01 7c51 yyyy yyyy
                         xxxx xxxx 0800 5832 6d04 0100 dde5 c339
                         8383 0d00
00:27:58.155627 ppp0 < x.x.x.x > y.y.y.y : icmp: echo reply (DF) (ttl
236, id 41987)
                         4500 0024 a403 4000 ec01 debf xxxx xxxx
                         yyyy yyyy 0000 6032 6d04 0100 dde5 c339
                         8383 0d00
```

The following ICMP Echo Request sent from my machine to the queried HP-UX 11.x just milliseconds after my reply to the HP-UX's query was sent. It has resulted in an ICMP Echo reply coming back from the queried machine. This time the DF bit was set with the ICMP Echo reply. Rather than sending an ICMP datagram that will be fragmented somewhere along the way to the destination machine, it is more beneficial from performance perspective, to fragment the ICMP datagram on sending. Setting the DF bit on the following replies would help to maintain the PMTU between the two systems, if for any reason, the PMTU would be decreased. For example, because the datagram have used another route to the destined system.

Sending immediately another ICMP Query message type to this particular HP-UX 11.x operating system based machine, will not result in the PMTU discovery process to be repeated. The DF Bit would be set within the ICMP Query reply. Expect a threshold to be maintained by the HP-UX 11.x. When reached the next time we query this host with any type of communication, the process of determining the PMTU using ICMP Echo request will begin again.

Why this method is bound to failure?

- Some ISPs would configure their routers not to allow fragmented ICMP datagrams through. I have encountered this behavior with different ISPs I have used.
- Some machines would be configured not to reply for an ICMP Echo requests coming from the Internet (if you read all of this research paper you'll do that).
- This ability can be used for a denial-of-service attack with the HP-UX 10.30, and/or 11.0x machines used as an amplifier for these attacks. Infect, HP has released a security bulletin dated February 13, 2000 about some issues regarding this PMTU discovery capability with ICMP. The bulletin states that "Depending upon the amount and nature of the inbound traffic, an HP-UX 10.30/11.00/11.04 system can be used to flood a target system with IP packets which could result in a denial of service"[30].
- Easy identification of HP-UX 10.30, 11.0x machines that had the default behavior not changed.

This gives us the ability to distinguish between Sun Solaris machines, HP-UX 11.0x/10.30 machines, and AIX 4.3.x based machines.

Sun Solaris sets the DF bit with the ICMP Query replies the operating system answers for, in order to support its global PMTU discovery process. If the networking link will not let the ICMP Query reply to get back to the querying host, because the MTU used is higher than the allowed

[30] HP Security Bulletin - "Security Vulnerability with PMTU strategy (revised)". February 13, 2000.

and fragmentation is not allowed (the DF Bit is set), than the size of the MTU used should be lowered. There is no active measures with Sun Solaris as far as I know.

This is a simple operating system fingerprinting method, which does not require additional or unusual patterns to be set.

The following operating systems where queries and checked for this kind of behavior:
Linux Kernel 2.4 test 2,4,5,6; Linux Kernel 2.2.x; FreeBSD 4.0, 3.4; OpenBSD 2.7,2.6; NetBSD 1.4.1,1.4.2; BSDI BSD/OS 4.0,3.1; Solaris 2.6,2.7,2.8; HP-UX 10.20, 11.0x; Compaq Tru64 5.0; Aix 4.1,3.2; Irix 6.5.3, 6.5.8; Ultrix 4.2 – 4.5; OpenVMS v7.1-2; Novel Netware 5.1 SP1, 5.0, 3.12; Microsoft Windows 98/98SE/ME, Microsoft Windows NT WRKS SP6a, Microsoft Windows NT Server SP4, Microsoft Windows 2000 Family.

6.2.1 Avoidance

With Sun Solaris and HPUX operating systems we can use a configuration option in order not to use the DF bit with the ICMP Query replies[31]. With HP-UX 10.30 and 11.x it would not allow the Path MTU Discovery process with ICMP Query replies to be done. This would avoid the fingerprinting method I have introduced, which is based on the fact the DF bit is set on ICMP Query replies from Sun Solaris, and HP-UX 10.30, and 11.x.

With HP-UX 10.30, & 11.0[32], one of the ndd command option is the ip_pmtu_strategy. The variable settings for this option are either 1 or 2. If this bit value is 2, than the Path MTU Discovery Process is used with ICMP Echo Requests. This is the default value. If this bit value equals 1, than the HP-UX machines will not use the ICMP echo-request PMTU discovery strategy, and will not set the DF bit after determining the accurate PMTU.

To turn off ip_path_mtu_discovery on a Sun Solaris machine use the following command as root:

```
# ndd -set  /dev/ip  ip_path_mtu_discovery 0
```

Than when the ICMP Echo Reply is sent (this example) the DF bit is not set:

```
# SING v1.0beta7 initiated on Host_Address at Thu Sep 14 10:01:02 2000
# Command line:
# -> sing -c 1 -L Host_Address
SINGing to Host_Address (IP_Address): 16 data bytes
16 bytes from 10.13.57.20: icmp_seq=0 ttl=254 TOS=0 time=1.578 ms

--- Host_Address sing statistics ---
1 packets transmitted, 1 packets received, 0% packet loss
round-trip min/avg/max = 1.578/1.578/1.578 ms
# SING finished at Thu Sep 14 10:01:02 2000
```

This was tested against Solaris 2.5.1, Solaris 2.6 and Solaris 2.7, all SPARC boxes.

Beware - With Sun Solaris turning this option off, will turn off the PMTU discovery process with TCP as well. This is not recommended because of performance issues.

[31] I do not have any information regarding AIX.
[32] Building a Bastion Host Using HP UX 11, Kevin Stevens, http://people.hp.se/stevesk/bastion11.html.

6.3 The IP Time-to-Live Field Value with ICMP

The sender sets the time to live field to a value that represents the maximum time the datagram is allowed to travel on the Internet.

The field value is decreased at each point that the Internet header is being processed. RFC 791 states that this field decreasement reflects the time spent processing the datagram. The field value is measured in units of seconds. The RFC also states that the maximum time to live value can be set to 255 seconds, which equals 4.25 minutes. The datagram must be discarded if this field value equals zero - before reaching its destination.

Relating to this field as a measure to assess time is a bit misleading. Some routers may process the datagram faster than a second, and some may process the datagram longer than a second.

The real intention is to have an upper bound to the datagrams lifetime, so infinite loops of undelivered datagrams will not jam the Internet.

Having a bound to the datagram's lifetime help us to prevent old duplicates to arrive after a certain time elapsed. So when we retransmit a piece of information which was not previously delivered we can be assured that the older duplicate is already discarded and will not interfere with the process.

The IP TTL field value with ICMP has two separate values, one for ICMP query messages and one for ICMP query replies.

The TTL field value helps us identify certain operating systems and groups of operating systems. It also provides us with the simplest means to add another check criteria when we are querying other host(s) or listening to traffic (sniffing).

6.3.1 IP TTL Field Value with ICMP Query Replies

We can use the IP TTL field value with the ICMP Query Reply datagrams to identify certain groups of operating systems. The method discussed in this section is a very simple one. We send an ICMP Query request message to a host. If we receive a reply, we would be looking at the IP TTL field value in the ICMP query reply. The next table describes the IP TTL field values with ICMP Echo replies for various operating systems. According to the table we can distinguish between certain operating systems:

Operating System	IP TTL on ICMP datagrams - In Reply -
LINUX Kernel 2.4	255
Kernel 2.2.14	255
Kernel 2.0.x[33]	64
FreeBSD 4.0	255
FreeBSD 3.4	255
OpenBSD 2.7	255
OpenBSD 2.6	255
NetBSD	255
BSDI BSD/OS 4.0	255
BSDI BSD/OS 3.1	255

[33] Stephane Omnes provided information about LINUX Kernel 2.0.x.

Operating System	IP TTL on ICMP datagrams - In Reply -
Solaris 2.5.1	255
Solaris 2.6	255
Solaris 2.7	255
Solaris 2.8	255
HP-UX v10.20	255
HP-UX v11.0	255
Compaq Tru64 v5.0	64
Irix 6.5.3	255
Irix 6.5.8	255
AIX 4.1	255
AIX 3.2	255
ULTRIX 4.2 – 4.5	255
OpenVMS v7.1-2	255
Windows 95	32
Windows 98	128
Windows 98 SE	128
Windows ME	128
Windows NT 4 WRKS SP 3	128
Windows NT 4 WRKS SP 6a	128
Windows NT 4 Server SP4	128
Windows 2000 Family	128

Table 5: IP TTL Field Values in replies from Various Operating Systems

If we would look at the ICMP Echo replies IP TTL field values than we can identify a few patterns:

- UNIX and UNIX-like operating systems use 255 as their IP TTL field value with ICMP query replies.
- Compaq Tru64 5.0 and LINUX 2.0.x are the exception, using 64 as its IP TTL field value with ICMP query replies.
- Microsoft Windows operating system based machines are using the value of 128.
- Microsoft Windows 95 is the only Microsoft operating system to use 32 as its IP TTL field value with ICMP query messages, making it unique among all other operating systems as well.

With the ICMP query replies we have an operating system that is clearly distinguished from the other - Windows 95. Other operating systems are grouped into the " 64 group" (LINUX based Kernel 2.0.x machines & Compaq Tru64 5.0), the "255 group" (UNIX and UNIX-like), and into the "128 group" (Microsoft operating systems).

We are not limited to ICMP ECHO replies only. We can use the other ICMP Query message types as well, and the results should be the same. In the next example an ICMP Timestamp request is sent to a Redhat 6.1 LINUX, Kernel 2.2.12 machine:

```
[root@stan /root]# icmpush -tstamp 192.168.5.5
kenny.sys-security.com -> 13:48:07
```

The Snort Trace:

```
01/26-13:51:29.342647 192.168.5.1 -> 192.168.5.5
ICMP TTL:254 TOS:0x0 ID:13170
TIMESTAMP REQUEST
88 16 D8 D9 02 8B 63 3D 00 00 00 00 00 00 00 00  ......c=........

01/26-13:51:29.342885 192.168.5.5 -> 192.168.5.1
ICMP TTL:255 TOS:0x0 ID:6096
TIMESTAMP REPLY
88 16 D8 D9 02 8B 63 3D 02 88 50 18 02 88 50 18  ......c=..P...P.
2A DE 1C 00 A0 F9                                *.....
```

IP TTL field value is 255 (the machine is on the same LAN).

We can use this information with other tests as, to provide us extra criteria with zero effort.

6.3.2 IP TTL Field Value with ICMP ECHO Requests

The examination of the IP TTL field value is not limited to ICMP Query replies only. We can learn a lot from the ICMP requests as well. The following is a Table summarizing various operating system default values for the IP TTL field embedded inside an ICMP Query request:

Operating System	IP TTL on ICMP datagrams - In Reply -	IP TTL on ICMP datagrams - In Req. -
Debian GNU/ LINUX 2.2, Kernel 2.4 test 2	255	64
Redhat LINUX 6.2 Kernel 2.2.14	255	64
LINUX Kernel 2.0.x	64	64
FreeBSD 4.0	255	255
FreeBSD 3.4	255	255
OpenBSD 2.7	255	255
OpenBSD 2.6	255	255
NetBSD	255	
BSDI BSD/OS 4.0	255	
BSDI BSD/OS 3.1	255	
Solaris 2.5.1	255	255
Solaris 2.6	255	255
Solaris 2.7	255	255
Solaris 2.8	255	255
HP-UX v10.20	255	255
HP-UX v11.0	255	
Compaq Tru64 v5.0	64	
Irix 6.5.3	255	
Irix 6.5.8	255	
AIX 4.1	255	
AIX 3.2	255	
ULTRIX 4.2 – 4.5	255	

Operating System	IP TTL on ICMP datagrams - In Reply -	IP TTL on ICMP datagrams - In Req. -
OpenVMS v7.1-2	255	
Windows 95	32	32
Windows 98	128	32
Windows 98 SE	128	32
Windows ME	128	32
Windows NT 4 WRKS SP 3	128	32
Windows NT 4 WRKS SP 6a	128	32
Windows NT 4 Server SP4	128	32
Windows 2000 Professional	128	128
Windows 2000 Server	128	128

Table 6: IP TTL Field Values in requests from Various Operating Systems

The ICMP Query message type used was ICMP Echo request, which is common on all operating systems tested using the ping utility.

- LINUX Kernel 2.0.x, 2.2.x & 2.4.x use 64 as their IP TTL Field Value with ICMP Echo Requests.
- FreeBSD 4.1, 4.0, 3.4; Sun Solaris 2.5.1, 2.6, 2.7, 2.8; OpenBSD 2.6, 2.7, NetBSD and HP UX 10.20 use 255 as their IP TTL field value with ICMP Echo requests. With the OSs listed above the same IP TTL Field value with any ICMP message is given.
- Windows 95/98/98SE/ME/NT4 WRKS SP3,SP4,SP6a/NT4 Server SP4 - all using 32 as their IP TTL field value with ICMP Echo requests.
- A Microsoft window 2000 is using 128 as its IP TTL Field Value with ICMP Echo requests.

We can distinguish between LINUX, Microsoft Windows 2000, the other Microsoft operating systems group, and the "255 group" using this method.

6.3.3 Correlating the Information

Using the IP TTL field value with ICMP messages we can distinguish between Microsoft Windows 2000, certain Microsoft Windows Operating systems, LINUX Kernel 2.2.x & 2.4.x, LINUX Kernel 2.0.x, and the *BSD and Solaris group.

Operating System	IP TTL value in the ECHO Requests	IP TTL value in the ECHO Replies
Microsoft Windows Family	32	128
*BSD and Solaris	255	255
LINUX Kernel 2.2.x & 2.4.x	64	255
LINUX Kernel 2.0.x	64	64
Microsoft Windows 2000	128	128
Microsoft Windows 95	33	32

Table 7: Further dividing the groups of operating systems according to IP TTL field value in the ICMP ECHO Requests and in the ICMP ECHO Replies

One would expect that the IP TTL field value would be the same with both ICMP Query requests and ICMP Query replies. Apparently this is not true and provide us with valuable information about the operating system querying / being queried.

6.4 Using Fragmented ICMP Address Mask Requests (Identifying Sun Solaris & HP-UX 11.0x machines)[34]

It appears that only some of the operating systems would answer an ICMP Address Mask Request as it is outlined in Table 2 in section 2.5. Those operating systems include - ULTRIX OpenVMS, Windows 95/98/98 SE/ME, NT below SP 4, HP-UX 11.0x and SUN Solaris. How can we distinguish between those who answer the request?

This is a regular ICMP Address Mask Request sent by SING to a SUN Solaris 2.7 machine:

```
[root@aik icmp]# ./sing -mask IP_Address
SINGing to IP_Address (IP_Address): 12 data bytes
12 bytes from IP_Address: icmp_seq=0 ttl=236 mask=255.255.255.0
12 bytes from IP_Address: icmp_seq=1 ttl=236 mask=255.255.255.0
12 bytes from IP_Address: icmp_seq=2 ttl=236 mask=255.255.255.0
12 bytes from IP_Address: icmp_seq=3 ttl=236 mask=255.255.255.0
12 bytes from IP_Address: icmp_seq=4 ttl=236 mask=255.255.255.0

--- IP_Address sing statistics ---
5 packets transmitted, 5 packets received, 0% packet loss
```

All operating systems that would answer with ICMP Address Mask Reply would reply with the Address Mask of the network they reside on.

What would happen if we would introduce a little twist? Lets say we would send those queries fragmented?

In the next example, I have sent ICMP Address Mask Request to the same SUN Solaris 2.7 box, this time fragmented to pieces of 8 bytes of IP data. As we can see the answer I got was unusual:

```
[root@aik icmp]# ./sing -mask -c 2 -F 8 IP_Address
SINGing to IP_Address (IP_Address): 12 data bytes
12 bytes from IP_Address: icmp_seq=0 ttl=241 mask=0.0.0.0
12 bytes from IP_Address: icmp_seq=1 ttl=241 mask=0.0.0.0

--- IP_Address sing statistics ---
2 packets transmitted, 2 packets received, 0% packet loss
[root@aik icmp]#
```

The tcpdump trace:

```
20:02:48.441174 ppp0 > y.y.y.y > Host_Address: icmp: address mask
request (frag 13170:8@0+)
                         4500 001c 3372 2000 ff01 50ab yyyy yyyy
                         xxxx xxxx 1100 aee3 401c 0000
20:02:48.442858 ppp0 > y.y.y.y > Host_Address: (frag 13170:4@8)
                         4500 0018 3372 0001 ff01 70ae yyyy yyyy
                         xxxx xxxx 0000 0000
```

[34] The Solaris portion was also discovered by Alfredo Andres Omella.

```
20:02:49.111427 ppp0 < Host_Address > y.y.y.y: icmp: address mask is
0x00000000 (DF)
                          4500 0020 3618 4000 f101 3c01 xxxx xxxx
                          yyyy yyyy 1200 ade3 401c 0000 0000 0000

20:02:49.441492 ppp0 > y.y.y.y > Host_Address: icmp: address mask
request (frag 13170:8@0+)
                          4500 001c 3372 2000 ff01 50ab yyyy yyyy
                          xxxx xxxx 1100 ade3 401c 0100
20:02:49.442951 ppp0 > y.y.y.y > Host_Address: (frag 13170:4@8)
                          4500 0018 3372 0001 ff01 70ae yyyy yyyy
                          xxxx xxxx 0000 0000
20:02:50.011433 ppp0 < Host_Address > y.y.y.y: icmp: address mask is
0x00000000 (DF)
                          4500 0020 3619 4000 f101 3c00 xxxx xxxx
                          yyyy yyyy 1200 ace3 401c 0100 0000 0000
```

The same SUN Solaris box now replies with a 0.0.0.0 as the Address Mask for the Network it resides on. The same behavioral patterns were produced against an HP-UX 11.0x operating system based machine[35].

What would happen with the other operating systems?

They all would respond with the real Address Mask in their replies.

Here we got a distinction between SUN Solaris & HP-UX 11.0x based machines to the other operating systems that would answer those queries.

In Section 6.2 we were discussing the various issues regarding the DF Bit usage within the ICMP Query replies. Both SUN Solaris and HP-UX 11.0x set the DF Bit by default in their ICMP Query replies. HP-UX 11.0x based machines starts setting the DF Bit after they finishes the ICMP based PMTU Discovery process (Querying the Host that has queried them with ICMP Echo request) – if it is enabled by default. This gives us another means to distinguish between those two operating systems on top of another method.

We can further try to distinguish between the remaining operating systems. This, if we would use the I=0 code method I am going to introduced in section 6.8:

Important notice: When I have tested this method I have encountered some problems replicating the results with different ISPs. As it seems from analyzing the information I got, certain ISPs would block fragmented ICMP datagrams. This behavior would not enable this method to succeed. One way of testing this is to send a regular ICMP Echo request. We should watch for a response from the probed machine. If received, than we should send ICMP Echo request, this time fragmented. If no reply is received than your ISP is blocking ICMP fragments probably.

[35] When I have published this information in Bugtraq (August 5, 2000) Peter J. Holzer notified me that HP-UX 11.00 produce the same behavior as the SUN Solaris boxes. Darren Reed also noted that because SUN Solaris and HP-UX 11.0 share the same third party (Mentat) implementation for some of their TCP/IP stacks this behavior is produced by both.

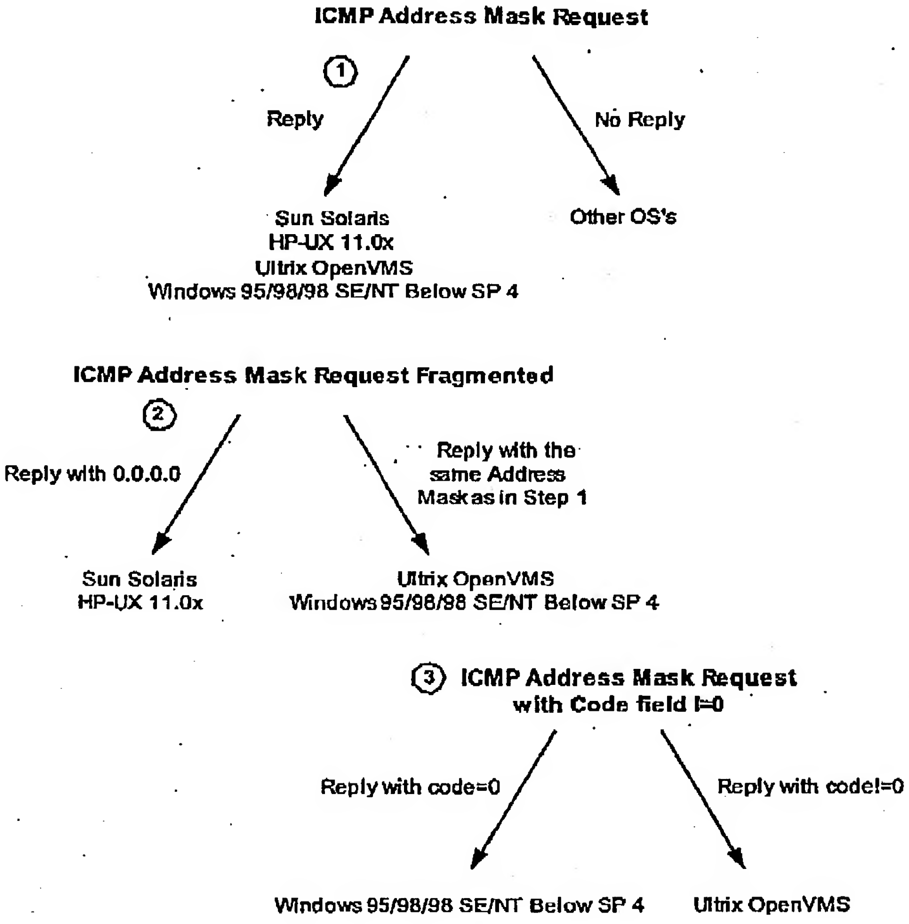

ICMP Address Mask Request
1
Reply
No Reply
Sun Solaris
HP-UX 11.0x
Ultrix OpenVMS
Windows 95/98/98 SE/NT Below SP 4
Other OS's
ICMP Address Mask Request Fragmented
2
Reply with 0.0.0.0
Reply with the same Address Mask as in Step 1
Sun Solaris
HP-UX 11.0x
Ultrix OpenVMS
Windows 95/98/98 SE/NT Below SP 4
3
ICMP Address Mask Request with Code field !=0
Reply with code=0
Reply with code!=0
Windows 95/98/98 SE/NT Below SP 4
Ultrix OpenVMS

Diagram 5: Finger Printing Using ICMP Address Mask Requests

Using Crafted ICMP Query Messages

Playing with the TOS Field

Each IP Datagram has an 8-bit field called the "TOS Byte", which represents the IP support for prioritization and Type-of-Service handling.

0	1	2	3	4	5	6	7
Precedence			TOS				MBZ

Figure 11: The Type of Service Byte

The "TOS Byte" consists of three fields.

The "Precedence field", which is 3-bit long, is intended to prioritize the IP Datagram. It has eight levels of prioritization[36]:

Precedence	Definition
0	Routine (Normal)
1	Priority
2	Immediate
3	Flash
4	Flash Override
5	Critical
6	Internetwork Control
7	Network control

Table 8: Precedence Field Values

Higher priority traffic should be sent before lower priority traffic.

The second field, 4 bits long, is the "Type-of-Service" field. It is intended to describe how the network should make tradeoffs between throughput, delay, reliability, and cost in routing an IP Datagram.

RFC 1349[37] has defined the "Type-of-Service" field as a single enumerated value, thus interpreted as a numeric value rather than independent flags (with RFC 791 the 4 bits were distinct options, allowing combinations as well). The 4 bits represents a maximum of 16 possible values.

Value (Hex)	Value (Dec)	Value (Binary)	Service
0	0	0000	Normal
1	1	1000	Minimize Delay
2	2	0100	Maximize Throughput
4	4	0010	Maximize Reliability
8	8	0001	Minimize Cost
F	15	1111	Maximize Security[38]

Table 9: Type-of-Service Field Values

What about the other 10 value possibilities?

RFC 1349 refer to this issue and states that "although the semantics of values other than the five listed above are not defined by this memo, they are perfectly legal TOS values, and hosts and routers must not preclude their use in any way"..."A host or a router need not make any distinction between TOS values who's semantics are defined by this memo and those that are not".

[36] RFC 791 – Internet Protocol, http://www.ietf.org/rfc/rfc791.txt.

[37] RFC 1349 - Type of Service in the Internet Protocol Suite, http://www.ietf.org/rfc/rfc1349.txt.

[38] RFC 1455 - Physical Link Security Type of Service, http://www.ietf.org/rfc/rfc1455.txt.

The last field, the "MBZ" (must be zero), is unused and must be zero. Routers and hosts ignore this last field. This field is 1 bit long.

Combining Type-of-Service flags with the different prioritization values, dictates very explicit types of behavior with certain types of data.

Please note the not all TCP/IP implementations would use this values (nor offer a mechanism for setting those values) and some will not handle datagrams which have Type-of-Service and/or Precedence values other than the defaults, differently.

6.5 Precedence Bits Echoing (Fingerprinting Microsoft Windows 2000, ULTRIX, HPUX 11.0&10.30, OpenVMS and more)

The precedence bits behavior is a problem. RFC 1122, which defines the requirements for Internet Hosts, does not outline the way to handle the Precedence Bits with ICMP. The RFC only statement about the Precedence Bits is:

> "The Precedence field is intended for Department of Defense applications of the Internet protocols. The use of non-zero values in this field is outside the scope of this document and the IP standard specification. Vendors should consult the Defense Communication Agency (DCA) for guidance on the IP Precedence field and its implications for other protocol layers. However, vendors should note that the use of precedence will most likely require that its value be passed between protocol layers in just the same way as the TOS field is passed".

This does not give us something to work with.

RFC 1812, Requirements for IP version 4 routers state that:

> "An ICMP reply message MUST have its IP Precedence field set to the value as the IP Precedence field in the ICMP request that provoked the reply".

Echoing back the Precedence field value has its logic, because the TOS field should be echoed back with an ICMP Query replies, and both the Precedence field and the TOS field were to dictate very explicit types of behavior with certain types of data.

As you can see we do not have a clear ruling about this issue. I was thinking it might be a ground for an operating system fingerprinting method.

Most operating systems I have checked will behave as the next behavioral example with AIX 4.3. With this example an ICMP Echo request is sent which carries a value for the TOS field.

```
[root@godfather precedence_echo]# /usr/local/bin/sing -c 5 -TOS 128
y.y.y.y
SINGing to y.y.y.y (y.y.y.y): 16 data bytes
16 bytes from y.y.y.y: seq=0 ttl=239 TOS=128 time=5896.472 ms
16 bytes from y.y.y.y: seq=1 ttl=239 TOS=128 time=5952.071 ms
16 bytes from y.y.y.y: seq=2 ttl=239 TOS=128 time=6102.020 ms
16 bytes from y.y.y.y: seq=3 ttl=239 TOS=128 time=6261.997 ms
16 bytes from y.y.y.y: seq=4 ttl=239 TOS=128 time=5842.726 ms

--- y.y.y.y sing statistics ---
5 packets transmitted, 5 packets received, 0% packet loss
```

```
round-trip min/avg/max = 5842.726/6011.057/6261.997 ms
[root@godfather precedence_echo]#
```

The tcpdump trace:

```
21:02:53.241666 ppp0 > x.x.x.x > y.y.y.y: icmp: echo request [tos 0x80]
(ttl 255, id 13170)
                         4580 0024 3372 0000 ff01 619c xxxx xxxx
                         yyyy yyyy 0800 c278 6f05 0000 dd97 0d3a
                         d8af 0300

21:02:59.134297 ppp0 < y.y.y.y > x.x.x.x: icmp: echo reply [tos 0x80]
(ttl 239, id 40656)
                         4580 0024 9ed0 0000 ef01 063e yyyy yyyy
                         xxxx xxxx 0000 ca78 6f05 0000 dd97 0d3a
                         d8af 0300
```

The Host queried is using the value used for the ICMP Echo Request with its ICMP Echo Reply.

Some operating systems are the exception.

The next example is with Microsoft Windows 2000. The same ICMP Echo Request was sent.

```
[root@godfather precedence_echo]# /usr/local/bin/sing -c 5 -TOS 128
y.y.y.y
SINGing to y.y.y.y (y.y.y.y): 16 data bytes
16 bytes from y.y.y.y: seq=0 ttl=111 TOS=0 time=6261.043 ms
16 bytes from y.y.y.y: seq=1 ttl=111 TOS=0 time=6422.019 ms
16 bytes from y.y.y.y: seq=2 ttl=111 TOS=0 time=6572.675 ms
16 bytes from y.y.y.y: seq=4 ttl=111 TOS=0 time=6282.022 ms

--- y.y.y.y sing statistics ---
5 packets transmitted, 4 packets received, 20% packet loss
round-trip min/avg/max = 6261.043/6384.440/6572.675 ms
[root@godfather precedence_echo]#
```

The tcpdump trace:

```
20:13:36.717070 ppp0 > x.x.x.x > y.y.y.y: icmp: echo request [tos 0x80]
(ttl 255, id 13170)
                         4580 0024 3372 0000 ff01 d95d xxxx xxxx
                         yyyy yyyy 0800 df43 c304 0000 508c 0d3a
                         edf0 0a00

20:13:42.974295 ppp0 < y.y.y.y > x.x.x.x: icmp: echo reply (ttl 111, id
26133)
                         4500 0024 6615 0000 6f01 373b yyyy yyyy
                         xxxx xxxx 0000 e743 c304 0000 508c 0d3a
                         edf0 0a00
```

The ICMP Echo Reply will not use the value assigned to the Precedence Bits with the ICMP Echo Request.

Which operating systems share this behavioral pattern? Microsoft Windows 2000 Family, and ULTRIX.

Differentiating between Microsoft Windows 2000 and Ultrix is easily achieved if we examine the IP TTL field value. With ULTRIX the value assigned to the ICMP Echo reply will be 255, with Microsoft Windows 2000 it will be 128.

Another interesting case is with HPUX 11.0. Lets examine the trace and logs:

```
[root@godfather precedence_echo]# /usr/local/bin/sing -c 2 -TOS 128
y.y.y.y
SINGing to y.y.y.y (y.y.y.y): 16 data bytes
16 bytes from y.y.y.y: seq=0 ttl=242 TOS=128 time=639.274 ms
16 bytes from y.y.y.y: seq=1 DF! ttl=242 TOS=0 time=310.427 ms

--- y.y.y.y sing statistics ---
2 packets transmitted, 2 packets received, 0% packet loss
round-trip min/avg/max = 310.427/474.850/639.274 ms
```

The first reply from the HPUX machine echoed back the TOS field value we were using with the ICMP Echo Request. But what have happened between the first and the second reply?

```
00:35:09.315260 ppp0 > x.x.x.x > y.y.y.y: icmp: echo request [tos 0x80]
(ttl 255, id 13170)
                     4580 0024 3372 0000 ff01 4bd1 xxxx xxxx
                     yyyy yyyy 0800 16f0 db3c 0000 9dc9 0d3a
                     56cf 0400

00:35:09.944274 ppp0 < y.y.y.y > x.x.x.x: icmp: echo request (DF) (ttl
242, id 22417)
                     4500 05dc 5791 4000 f201 ef79 yyyy yyyy
                     xxxx xxxx 0800 7e52 9abc def0 0000 0000
                     0000 0000 0000 0000 0000 0000 0000 0000
                     0000 0000 0000 0000 0000 0000 0000 0000
                     0000 0000 0000 0000 0000 0000 0000 0000
                     0000 0000 0000 0000 0000 0000 0000 0000
                     0000 0000 0000 0000 0000 0000 0000 0000
                     0000 0000 0000 0000 0000 0000 0000 0000
                     0000 0000 0000 0000 0000 0000 0000 0000
                     0000 0000 0000 0000 0000 0000 0000 0000
                     0000 0000 0000 0000 0000 0000 0000 0000
                     0000 0000 0000 0000 0000 0000 0000 0000
                     0000 0000 0000 0000 0000 0000 0000 0000
                     0000 0000 0000 0000 0000 0000 0000 0000
                     0000 0000 0000 0000 0000 0000 0000 0000
                     0000 0000 0000 0000 0000 0000 0000 0000
                     0000 0000 0000 0000 0000 0000 0000 0000
                     0000 0000 0000 0000 0000 0000 0000 0000
                     0000 0000 0000 0000 0000 0000 0000 0000
                     0000 0000 0000 0000 0000 0000 0000 0000
                     0000 0000 0000 0000 0000 0000 0000 0000
                     0000 0000 0000 0000 0000 0000 0000 0000
                     0000 0000 0000 0000 0000 0000 0000 0000
                     0000 0000 0000 0000 0000 0000 0000 0000
                     0000 0000 0000 0000 0000 0000 0000 0000
                     0000 0000 0000 0000 0000 0000 0000 0000
                     0000 0000 0000 0000 0000 0000 0000 0000
```

```
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
```

```
                    0000 0000 0000 0000 0000 0000 0000 0000
                    0000 0000 0000 0000 0000 0000 0000 0000
                    0000 0000 0000 0000 0000 0000 0000 0000
                    0000 0000 0000 0000 0000 0000 0000 0000
                    0000 0000 0000 0000 0000 0000 0000 0000
                    0000 0000 0000 0000 0000 0000 0000 0000
                    0000 0000 0000 0000 0000 0000 0000 0000
                    0000 0000 0000 0000 0000 0000 0000 0000
                    0000 0000 0000 0000 0000 0000 0000 0000
                    0000 0000 0000 0000 0000 0000
```

The first request was sent, as an instant reply the HPUX 11.0 machine started its PMTU discovery process with ICMP Echo Requests and sent an ICMP Echo Request 1500 bytes long[39].

```
00:35:09.944355 ppp0 > x.x.x.x > y.y.y.y: icmp: echo reply (ttl 255, id
14194)
                    4500 05dc 3772 0000 ff01 4299 xxxx xxxx
                    yyyy yyyy 0000 8652 9abc def0 0000 0000
                    0000 0000 0000 0000 0000 0000 0000 0000
                    0000 0000 0000 0000 0000 0000 0000 0000
                    0000 0000 0000 0000 0000 0000 0000 0000
                    0000 0000 0000 0000 0000 0000 0000 0000
                    0000 0000 0000 0000 0000 0000 0000 0000
                    0000 0000 0000 0000 0000 0000 0000 0000
                    0000 0000 0000 0000 0000 0000 0000 0000
                    0000 0000 0000 0000 0000 0000 0000 0000
                    0000 0000 0000 0000 0000 0000 0000 0000
                    0000 0000 0000 0000 0000 0000 0000 0000
                    0000 0000 0000 0000 0000 0000 0000 0000
                    0000 0000 0000 0000 0000 0000 0000 0000
                    0000 0000 0000 0000 0000 0000 0000 0000
                    0000 0000 0000 0000 0000 0000 0000 0000
                    0000 0000 0000 0000 0000 0000 0000 0000
                    0000 0000 0000 0000 0000 0000 0000 0000
                    0000 0000 0000 0000 0000 0000 0000 0000
                    0000 0000 0000 0000 0000 0000 0000 0000
                    0000 0000 0000 0000 0000 0000 0000 0000
                    0000 0000 0000 0000 0000 0000 0000 0000
                    0000 0000 0000 0000 0000 0000 0000 0000
                    0000 0000 0000 0000 0000 0000 0000 0000
                    0000 0000 0000 0000 0000 0000 0000 0000
                    0000 0000 0000 0000 0000 0000 0000 0000
                    0000 0000 0000 0000 0000 0000 0000 0000
                    0000 0000 0000 0000 0000 0000 0000 0000
                    0000 0000 0000 0000 0000 0000 0000 0000
                    0000 0000 0000 0000 0000 0000 0000 0000
                    0000 0000 0000 0000 0000 0000 0000 0000
                    0000 0000 0000 0000 0000 0000 0000 0000
                    0000 0000 0000 0000 0000 0000 0000 0000
                    0000 0000 0000 0000 0000 0000 0000 0000
                    0000 0000 0000 0000 0000 0000 0000 0000
                    0000 0000 0000 0000 0000 0000 0000 0000
```

[39] For an explanation about the HPUX 11.0 PMTU process using ICMP Echo Requests please see section 6.2.

```
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
```

```
                          0000 0000 0000 0000 0000 0000

00:35:09.954282 ppp0 < y.y.y.y > x.x.x.x: icmp: echo reply [tos 0x80]
(ttl 242, id 22418)
                          4580 0024 5792 0000 f201 34b1 yyyy yyyy
                          xxxx xxxx 0000 1ef0 db3c 0000 9dc9 0d3a
                          56cf 0400
```

The ICMP Echo Reply received from the HPUX 11.0 machine for the ICMP Echo Request echoed back the TOS field value.

Another ICMP Echo Request was sent with TOS field value of 0x80 hex:

```
00:35:10.314321 ppp0 > x.x.x.x > y.y.y.y: icmp: echo request [tos 0x80]
(ttl 255, id 13170)
                          4580 0024 3372 0000 ff01 4bd1 xxxx xxxx
                          yyyy yyyy 0800 b7f3 db3c 0100 9ec9 0d3a
                          b3cb 0400

00:35:10.624275 ppp0 < y.y.y.y > x.x.x.x: icmp: echo reply (DF) (ttl
242, id 22419)
                          4500 0024 5793 4000 f201 f52f yyyy yyyy
                          xxxx xxxx 0000 bff3 db3c 0100 9ec9 0d3a
                          b3cb 0400
```

The ICMP Echo Reply received did not echo back the TOS field value, and set the DF bit. The PMTU discovery process finished its initial stages and went to regular operation. From now on the ICMP Echo Replies did not echo the TOS field value.

This gives us the ability to track down HPUX 11.0 (and 10.30) machines when they are using the PMTU Discovery process.

6.5.1 Changed Pattern with other ICMP Query Message Types

We can identify change of pattern with OpenVMS, Windows 98, 98SE, and ME. With ICMP Echo replies they all would echo back the TOS field value, but with ICMP Timestamp replies they will change the behavior and send back 0x000. Since OpenVMS use 255 as its IP TTL field value, and the Microsoft Windows based machines use 128, we can differentiate between them and isolate OpenVMS, and the Microsoft based OSs.

Further distinction between the Microsoft operating systems can be achieved if we will query them with ICMP Address Mask request, which only Microsoft Windows 98/98SE will answer for. The Microsoft Windows ME will not reply, enabling us to identify it.

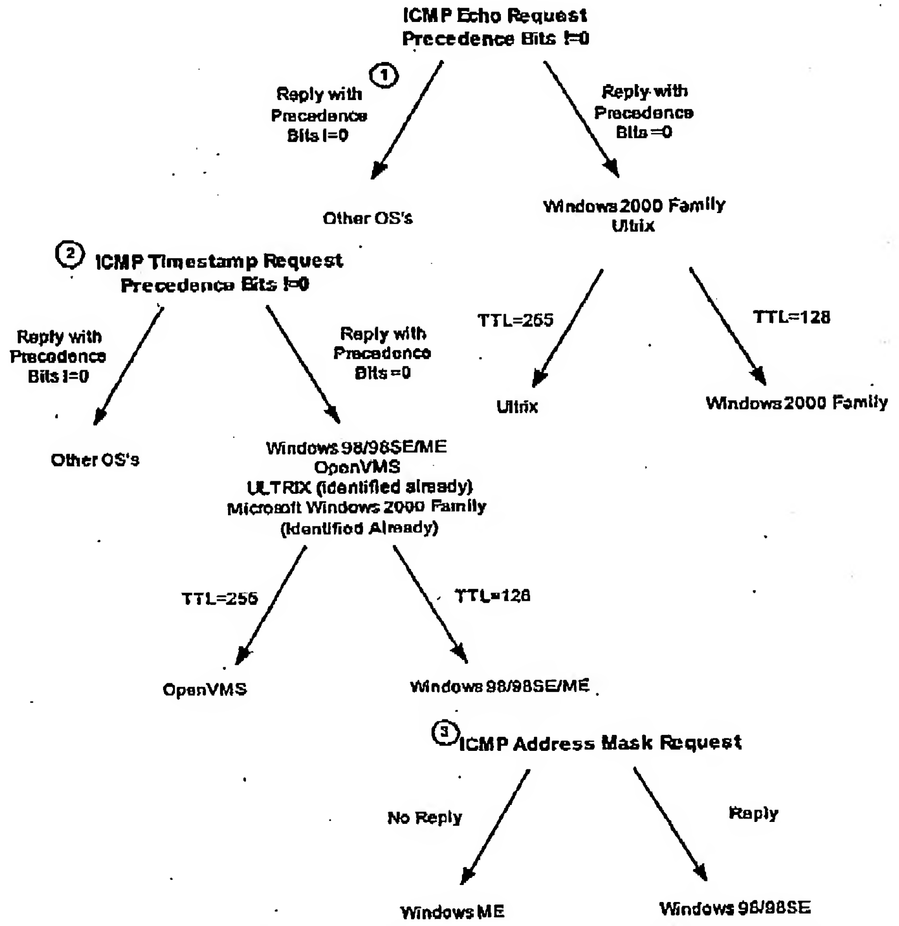


Diagram 6: An example for a way to fingerprint Microsoft Windows 2000, Ultrix, HPUX 11.0 & 10.30, OpenVMS, Microsoft Windows ME, and Microsoft Windows 98/98SE based machines with ICMP Query messages with the Precedence Bits field !=0

Operating System	Information Request With Precedence!=0	Time Stamp Request With Precedence!=0	Address Mask Request With Precedence!=0	Echo Request With Precedence!=0
Debian GNU/ LINUX 2.2, Kernel 2.4 test 2	Not Answering	!=0x00	Not Answering	!=0x00
Redhat LINUX 6.2 Kernel 2.2.14	Not Answering	!=0x00	Not Answering	!=0x00
FreeBSD 4.0	Not Answering	!=0x00	Not Answering	!=0x00
FreeBSD 4.1.1	Not Answering	!=0x00	Not Answering	
OpenBSD 2.7	Not Answering		Not Answering	!=0x00
OpenBSD 2.6	Not Answering		Not Answering	!=0x00
NetBSD	Not Answering		Not Answering	!=0x00
BSDI BSD/OS 4.0	Not Answering		Not Answering	!=0x00

Operating System	Information Request With Precedence!=0	Time Stamp Request With Precedence!=0	Address Mask Request With Precedence!=0	Echo Request With Precedence!=0
BSDI BSD/OS 3.1	Not Answering		Not Answering	!=0x00
Solaris 2.5.1	Not Implemented			
Solaris 2.6	Not Implemented	!=0x00	!=0x00	!=0x00
Solaris 2.7	Not Implemented	!=0x00	!=0x00	!=0x00
Solaris 2.8	Not Implemented	!=0x00	!=0x00	!=0x00
HP-UX v10.20			Not Answering	
HP-UX v11.0	Not Answering	Not Answering	!=0x00 -> 0x00	!=0x00 -> 0x00
Compaq Tru64 v5.0	!=0x00	!=0x00	Not Answering	!=0x00
AIX 4.3	!=0x00	!=0x00	Not Answering	!=0x00
AIX 4.2.1	!=0x00	!=0x00	Not Answering	!=0x00
AIX 4.1	!=0x00	!=0x00	Not Answering	!=0x00
AIX 3.2	!=0x00	!=0x00	Not Answering	!=0x00
ULTRIX 4.2 – 4.5	0x00	0x00	0x00	0x00
OpenVMS v7.1-2	0x00	0x00	0x00	!=0x00
Windows 95	Not Answering	Not Answering	0x00	!=0x00
Windows 98	Not Answering	0x00	0x00	!=0x00
Windows 98 SE	Not Answering	0x00	Not Answering	!=0x00
Windows ME	Not Answering	0x00		!=0x00
Windows NT 4 WRKS SP 3	Not Answering	Not Answering		!=0x00
Windows NT 4 WRKS SP 6a	Not Answering	Not Answering	Not Answering	
Windows NT 4 Server SP4	Not Answering	Not Answering	Not Answering	!=0x00
Windows 2000 Professional	Not Answering	0x00	Not Answering	0x00
Windows 2000 Server	Not Answering	0x00	Not Answering	0x00

Table 10: ICMP Query Message Types with Precedence Bits ! = 0

6.6 TOSing OSs out of the Window / "TOS Echoing" (Fingerprinting Microsoft Windows 2000)

6.6.1 The use of the Type-of-Service field with the Internet Control Message Protocol

RFC 1349 also define the usage of the Type-of-Service field with the ICMP messages. It distinguishes between ICMP error messages (Destination Unreachable, Source Quench, Redirect, Time Exceeded, and Parameter Problem), ICMP query messages (Echo, Router Solicitation, Timestamp, Information request, Address Mask request) and ICMP reply messages (Echo reply, Router Advertisement, Timestamp reply, Information reply, Address Mask reply).

Simple rules are defined:

- An ICMP error message is always sent with the default TOS (0x00)
- An ICMP request message may be sent with any value in the TOS field. "A mechanism to allow the user to specify the TOS value to be used would be a useful feature in many applications that generate ICMP request messages"[40].

[40] RFC 1349 - Type of Service in the Internet Protocol Suite, http://www.ietf.org/rfc/rfc1349.txt.

The RFC further specify that although ICMP request messages are normally sent with the default TOS, there are sometimes good reasons why they would be sent with some other TOS value.

- An ICMP reply message is sent with the same value in the TOS field as was used in the corresponding ICMP request message.

Using this logic I have decided to check if certain operating systems react correctly to an ICMP Query messages with a Type-of-Service field value, which is different than the default (0x00).

The check out was produced with all ICMP query message types sent with a Type-of-Service field set to a known value, than set to an unknown value (the term known and unknown are used here because I was not experimenting with non-legit values, and since any value may be sent inside this field).

The following example is an ICMP Echo request sent to my FreeBSD 4.0 machine with the TOS field equals an 8 hex value, which is a legit TOS value. The tool used here is SING[41]:

```
[root@godfather bin]# ./sing -echo -TOS 8 IP_Address
SINGing to IP_Address (IP_Address): 16 data bytes
16 bytes from IP_Address: icmp_seq=2 ttl=243 TOS=8 time=260.043 ms
16 bytes from IP_Address: icmp_seq=3 ttl=243 TOS=8 time=180.011 ms
16 bytes from IP_Address: icmp_seq=4 ttl=243 TOS=8 time=240.240 ms
16 bytes from IP_Address: icmp_seq=5 ttl=243 TOS=8 time=260.037 ms
16 bytes from IP_Address: icmp_seq=6 ttl=243 TOS=8 time=290.033 ms

--- IP_Address sing statistics ---
7 packets transmitted, 5 packets received, 28% packet loss
round-trip min/avg/max = 180.011/246.073/290.033 ms
[root@godfather bin]#
```

The tcpdump trace:

```
17:23:46.605297 if 4  > x.x.x.x > y.y.y.y: icmp: echo request [tos 0x8]
(ttl 255, id 13170)
                         4508 0024 3372 0000 ff01 60e4 xxxx xxxx
                         yyyy yyyy 0800 0e9a d604 0600 f2ea bc39
                         553c 0900
17:23:46.895255 if 4  < y.y.y.y > x.x.x.x: icmp: echo reply [tos 0x8]
(ttl 243, id 58832)
                         4508 0024 e5d0 0000 f301 ba85 yyyy yyyy
                         xxxx xxxx 0000 169a d604 0600 f2ea bc39
                         553c 0900
```

This is the second test I have produced, sending ICMP Echo request with the Type-of-Service field set to a 10 Hex value, a value that is not a known Type-of-Service value:

[41] SING has the ability to monitor for any replies and than print the received TOS value. I find this option very useful, and thank the author for embedding this function, as I requested.

```
[root@godfather bin]# ./sing -echo -TOS 10 IP_Address
SINGing to IP_Address (IP_Address): 16 data bytes
16 bytes from IP_Address: icmp_seq=0 ttl=243 TOS=10 time=197.933 ms
16 bytes from IP_Address: icmp_seq=1 ttl=243 TOS=10 time=340.048 ms
16 bytes from IP_Address: icmp_seq=2 ttl=243 TOS=10 time=250.025 ms
16 bytes from IP_Address: icmp_seq=3 ttl=243 TOS=10 time=230.019 ms
16 bytes from IP_Address: icmp_seq=4 ttl=243 TOS=10 time=270.017 ms
16 bytes from IP_Address: icmp_seq=5 ttl=243 TOS=10 time=270.017 ms
16 bytes from IP_Address: icmp_seq=6 ttl=243 TOS=10 time=260.021 ms

--- IP_Address sing statistics ---
7 packets transmitted, 7 packets received, 0% packet loss
round-trip min/avg/max = 197.933/259.726/340.048 ms
```

The tcpdump trace:

```
17:24:36.155298 if 4  > y.y.y.y > x.x.x.x: icmp: echo request [tos
0xa,ECT]  (ttl 255, id 13170)
                     450a 0024 3372 0000 ff01 60e2 yyyy yyyy
                     xxxx xxxx 0800 af77 d904 0600 24eb bc39
                     865e 0200
17:24:36.415254 if 4  < x.x.x.x > y.y.y.y: icmp: echo reply [tos
0xa,ECT]  (ttl 243, id 65031)
                     450a 0024 fe07 0000 f301 a24c xxxx xxxx
                     yyyy yyyy 0000 b777 d904 0600 24eb bc39
                     865e 0200
```

As it can be seen from the tcpdump trace, the ICMP Echo reply messages have maintained the Type-of-Service value as was used in the corresponding ICMP request message.

FreeBSD 4.0 does not respond to ICMP Information request, or to ICMP Address Mask requests. I had to verify with ICMP Timestamp requests with the same Type-of-Service values as with the previous ICMP Echo requests that this behavior is produced with ICMP Timestamp request and replies as well.

Again the tool I have used is SING:

```
[root@godfather bin]# ./sing -tstamp -TOS 8 IP_Address
SINGing to IP_Address (IP_Address): 20 data bytes
20 bytes from IP_Address: icmp_seq=0 ttl=243 TOS=8 diff=6832668
20 bytes from IP_Address: icmp_seq=1 ttl=243 TOS=8 diff=6832403
20 bytes from IP_Address: icmp_seq=2 ttl=243 TOS=8 diff=6832633
20 bytes from IP_Address: icmp_seq=3 ttl=243 TOS=8 diff=6832605
20 bytes from IP_Address: icmp_seq=4 ttl=243 TOS=8 diff=6832431

--- IP_Address sing statistics ---
5 packets transmitted, 5 packets received, 0% packet loss
[root@godfather bin]#
```

The tcpdump trace:

```
17:26:00.455295 if 4  > x.x.x.x > y.y.y.y: icmp: time stamp request
[tos 0x8]  (ttl 255, id 13170)
                     4508 0028 3372 0000 ff01 60e0 xxxx xxxx
```

```
                    yyyy yyyy 0d00 345b dd04 0400 0318 da87
                    0000 0000 0000 0000
17:26:00.755254 if 4  < y.y.y.y > x.x.x.x: icmp: time stamp reply [tos
0x8]  (ttl 243, id 5867)
                    4508 0028 16eb 0000 f301 8967 yyyy yyyy
                    xxxx xxxx 0e00 f4ec dd04 0400 0318 da87
                    0380 1bb7 0380 1bb7
```

The second test with TOS field value set to 10 Hex value:

```
[root@godfather bin]# ./sing -tstamp -TOS 10 IP_Address
SINGing to IP_Address (IP_Address): 20 data bytes
20 bytes from IP_Address: icmp_seq=0 ttl=243 TOS=10 diff=6766872
20 bytes from IP_Address: icmp_seq=1 ttl=243 TOS=10 diff=6767059
20 bytes from IP_Address: icmp_seq=2 ttl=243 TOS=10 diff=6767059
20 bytes from IP_Address: icmp_seq=3 ttl=243 TOS=10 diff=6767063
20 bytes from IP_Address: icmp_seq=4 ttl=243 TOS=10 diff=6766892
20 bytes from IP_Address: icmp_seq=5 ttl=243 TOS=10 diff=6766887
20 bytes from IP_Address: icmp_seq=6 ttl=243 TOS=10 diff=6766873
20 bytes from IP_Address: icmp_seq=7 ttl=243 TOS=10 diff=6767057
^C

--- 194.47.250.37 sing statistics ---
9 packets transmitted, 9 packets received, 0% packet loss
[root@godfather bin]#
```

The tcpdump trace:

```
17:25:42.548597 if 4  > x.x.x.x > y.y.y.y: icmp: time stamp request
[tos 0xa,ECT]  (ttl 255, id 13170)
                    450a 0028 3372 0000 ff01 60de xxxx xxxx
                    yyyy yyyy 0d00 7f4e dc04 0000 0318 9494
                    0000 0000 0000 0000
17:25:42.795254 if 4  < y.y.y.y > x.x.x.x: icmp: time stamp reply [tos
0xa,ECT]  (ttl 243, id 3519)
                    450a 0028 0dbf 0000 f301 9291 yyyy yyyy
                    xxxx xxxx 0e00 cbf6 dc04 0000 0318 9494
                    037f d5ac 037f d5ac
```

The same behavior was produced. The ICMP Timestamp replies were sent with the TOS field value equals the TOS filed value of the ICMP Timestamp requests.

Ok. I was curious again. I imagined that the Microsoft Windows implementation of the things might be a little different.

When I was examining ICMP Echo requests I noticed something is wrong with Microsoft:

```
[root@godfather bin]# ./sing -echo -TOS 8 Host_Address
SINGing to Host_Address (IP_Address): 16 data bytes
16 bytes from IP_Address: icmp_seq=0 ttl=113 TOS=0 time=278.813 ms
16 bytes from IP_Address: icmp_seq=1 ttl=113 TOS=0 time=239.935 ms
```

```
16 bytes from IP_Address: icmp_seq=2 ttl=113 TOS=0 time=249.937 ms
16 bytes from IP_Address: icmp_seq=3 ttl=113 TOS=0 time=229.962 ms
16 bytes from IP_Address: icmp_seq=4 ttl=113 TOS=0 time=249.951 ms

--- Host_Address sing statistics ---
5 packets transmitted, 5 packets received, 0% packet loss
round-trip min/avg/max = 229.962/249.720/278.813 ms
[root@godfather bin]#
```

The tcpdump trace:

```
17:28:08.346537 if 4  > x.x.x.x > y.y.y.y: icmp: echo request [tos 0x8]
(ttl 255, id 13170)
                         4508 0024 3372 0000 ff01 083f xxxx xxxx
                         yyyy yyyy 0800 cd8b e704 0000 f8eb bc39
                         8949 0500
17:28:08.625250 if 4  < y.y.y.y > x.x.x.x: icmp: echo reply (ttl 113,
id 12951)
                         4500 0024 3297 0000 7101 9722 yyyy yyyy
                         xxxx xxxx 0000 d58b e704 0000 f8eb bc39
                         8949 0500
```

Oops! Some one zero out my Type-of-Service field!

Before I would let you know who of all Microsoft Windows operating systems did that, I am going to list the Microsoft operating systems who behave correctly – Microsoft Windows 98/SE/ME, Microsoft Windows NT 4 Workstation SP3, Microsoft Windows NT 4 Server SP4, Microsoft Windows NT 4 Workstation SP6a.

The Microsoft Windows 2000 family (Professional, Server, Advanced Server) zero out this field on the ICMP Echo reply.

Is this makes those Microsoft Windows 2000 machines identified easily and uniquely?

99.9% yes. The other 0.01 % belongs to Ultrix & Novel Netware.

From the operating systems I have checked (Linux Kernel 2.2.x, Linux Kernel 2.4 test 2/4/5, FreeBSD 4.0 & 4.1, OpenBSD 2.6 & 2.7, NetBSD 1.4.2, SUN Solaris 2.7 & 2.8, Compaq Tru64 UNIX 5.0, AIX 4.1 & 3.2, OpenVMS v7.2, Irix 6.5.3 & 6.5.8, Ultrix 4.2-4.5, Microsoft Windows 98/SE/ME, Microsoft Windows NT 4 Workstation & Server with various service packs, Microsoft Windows 2000 Professional, Server & Advanced Server) only Ultrix & Novell Netware behaved like the Microsoft Windows 2000 machines.

How can we distinguish between those?
First, there are much fewer Ultrix and Novell operating systems based machines out there than Microsoft's Windows 2000 based machines (I see your faces – not convincing enough).

The fast track in distinguishing between Ultrix, Novell Netware and Microsoft Windows 2000 is a simple one. - By looking at the IP TTL field value within the ICMP Echo replies. Microsoft Windows 2000 family of operating systems use 128 as their default IP TTL field value in ICMP ECHO replies while Ultrix uses 255. The problem is Novell uses the same value for its IP TTL

Field value as Microsoft Windows 2000 based machines (For more information about the IP TTL field value see section 6.3 – „The IP Time-to-Live Field Value with ICMP").

As a next step we can use various types of queries in order to distinguish between the Novell Netware based machines, to the Microsoft Windows 2000 based machines. One of those steps can be an ICMP Timestamp request sent to the "suspicious" machines. No reply from one of the machines will indicate it may be using Novell Netware, a reply from a machine will indicate it is using one of Microsoft Windows 2000 operating system family product. More sophisticated ICMP queries could replace the one I have introduced.

Other ICMP query message types help us to identify a unique group of Microsoft operating systems. As a rule all operating systems except the named Microsoft windows operating systems here, maintain a single behavior regarding the Type-of-Service field. All would maintain the same values with different types of ICMP requests. But, again, Microsoft have some of the "top" people understanding TCP/IP to the degree we humans do not understand.

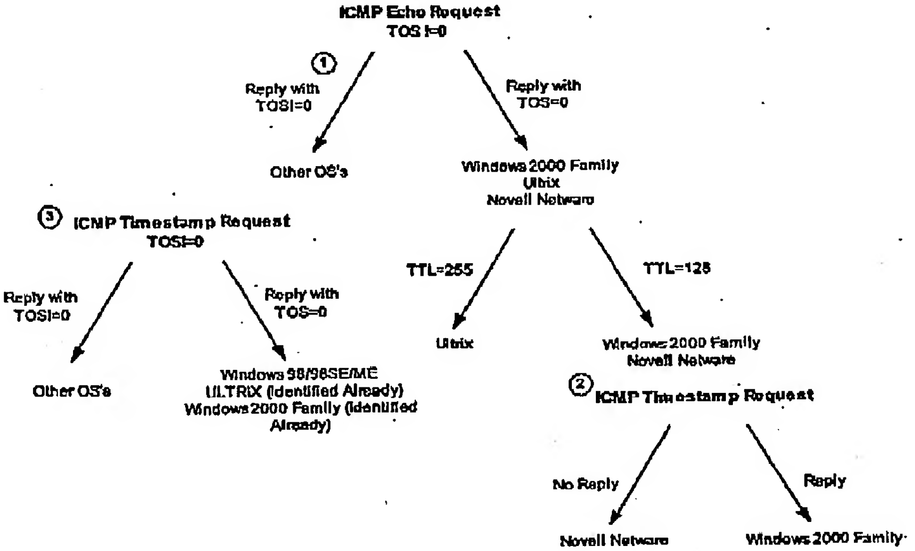


Diagram 7: An example for a way to fingerprint Windows 2000, Ultrix, and Novell Netware based machines with ICMP Query messages with the TOS bits field !=0

We have the following Microsoft operating systems zero out (0x00) the Type-of-Service field with the replies for ICMP Timestamp requests: Microsoft Windows 98/98SE/ME. Microsoft Windows 2000 machines would zero out the TOS field with ICMP Timestamp replies as well.

This means that Microsoft Windows 98/98SE/ME would not zero out the Type-of-Service field value with ICMP Echo requests but will do so with ICMP Timestamp requests.
With the introduced fingerprinting methods in this section we got a way to fingerprint Microsoft Windows 2000, Ultrix, and Novel Netware machines from the rest of the operating systems world. We have a way to distinguish Microsoft Windows 98/98SE/ME (and to set those apart) from the rest of the operating system world, as well.

Operating System	Information Request With TOS!=0x00	Time Stamp Request With TOS!=0x00	Address Mask Request With TOS!=0x00	Echo Request With TOS!=0x00
Debian GNU/ LINUX 2.2, Kernel 2.4 test 2	Not Answering	!=0x00	Not Answering	!=0x00
Redhat LINUX 6.2 Kernel 2.2.14	Not Answering	!=0x00	Not Answering	!=0x00
FreeBSD 4.0	Not Answering	!=0x00	Not Answering	!=0x00
FreeBSD 3.4	Not Answering		Not Answering	
OpenBSD 2.7	Not Answering	!=0x00	Not Answering	!=0x00
OpenBSD 2.6	Not Answering	!=0x00	Not Answering	!=0x00
NetBSD	Not Answering	!=0x00	Not Answering	!=0x00
BSDI BSD/OS 4.0	Not Answering	!=0x00	Not Answering	!=0x00
BSDI BSD/OS 3.1	Not Answering	!=0x00	Not Answering	!=0x00
Solaris 2.5.1	Not Implemented			
Solaris 2.6	Not Implemented			
Solaris 2.7	Not Implemented	!=0x00	!=0x00	!=0x00
Solaris 2.8	Not Implemented	!=0x00	!=0x00	!=0x00
HP-UX v10.20			Not Answering	
HP-UX v11.0	Not Answering	Not Answering	!=0x00	!=0x00
Compaq Tru64 v5.0		!=0x00	Not Answering	!=0x00
Irix 6.5.3	Not Answering	!=0x00	Not Answering	!=0x00
Irix 6.5.8	Not Answering	!=0x00	Not Answering	!=0x00
AIX 4.1		!=0x00	Not Answering	!=0x00
AIX 3.2		!=0x00	Not Answering	!=0x00
ULTRIX 4.2 – 4.5		0x00	0x00	0x00
OpenVMS v7.1-2		!=0x00	!=0x00	!=0x00
Novell Netware 5.1 SP1	Not Answering	Not Answering	Not Answering	0x00
Novell Netware 5.0	Not Answering	Not Answering	Not Answering	0x00
Novell Netware 3.12	Not Answering	Not Answering	Not Answering	0x00
Windows 95	Not Answering	Not Answering		
Windows 98	Not Answering	0x00	0x00	!=0x00
Windows 98 SE	Not Answering	0x00		!=0x00
Windows ME	Not Answering	0x00	Not Answering	!=0x00
Windows NT 4 WRKS SP 3	Not Answering	Not Answering		!=0x00
Windows NT 4 WRKS SP 6a	Not Answering	Not Answering	Not Answering	!=0x00
Windows NT 4 Server SP4	Not Answering	Not Answering	Not Answering	!=0x00
Windows 2000 Professional	Not Answering	0x00	Not Answering	0x00
Windows 2000 Server	Not Answering	0x00	Not Answering	0x00

Table 11: ICMP Query Message Types with TOS! = 0

6.7 Using the TOS Byte's Unused Bit (Fingerprinting Microsoft Windows 2000, ULTRIX and more)

RFC 1349 states that the last field of the TOS byte, the "MBZ" (must be zero), is unused and must be zero. The RFC also states that routers and hosts ignore the value of this bit.

This is the only statement about the unused bit in the TOS Byte in the RFCs. The RFC states: "The originator of a datagram sets this field to Zero".

Obviously it was meant that this field would be always zero. But what will happen if we would set this bit with our ICMP Echo Requests? Will this bit be zero out on reply or will it be echoed back?

Only with ICMP Echo requests we can have a clear identification of OSs.

The next example is an ICMP Echo Request sent with the TOS bit in the TOS Byte set, targeting a FreeBSD 4.1.1 machine:

```
[root@godfather /root]# /usr/local/bin/sing  -c 2 -TOS 1 y.y.y.y
SINGing to y.y.y.y (y.y.y.y): 16 data bytes
16 bytes from y.y.y.y: seq=0 ttl=233 TOS=1 time=330.461 ms
16 bytes from y.y.y.y: seq=1 ttl=233 TOS=1 time=723.300 ms

--- y.y.y.y sing statistics ---
2 packets transmitted, 2 packets received, 0% packet loss
round-trip min/avg/max = 330.461/526.880/723.300 ms
[root@godfather /root]#
```

Echoing back the Unused bit in the TOS Byte represents the behavior of most of the operating systems I have checked this method against.

Which operating systems are the exceptions?
The next example is with Microsoft Windows 2000 as the targeted machine:

```
[root@godfather precedence_echo]# /usr/local/bin/sing -c 2 -TOS 1
y.y.y.y
SINGing to y.y.y.y (y.y.y.y): 16 data bytes
16 bytes from y.y.y.y: seq=0 ttl=111 TOS=0 time=299.188 ms
16 bytes from y.y.y.y: seq=1 ttl=111 TOS=0 time=280.321 ms

--- y.y.y.y sing statistics ---
2 packets transmitted, 2 packets received, 0% packet loss
round-trip min/avg/max = 280.321/289.755/299.188 ms
[root@godfather precedence_echo]#
```

The tcpdump trace:

```
00:17:01.765492 ppp0 > x.x.x.x > y.y.y.y: icmp: echo request [tos 0x1]
(ttl 255, id 13170)
                         4501 0024 3372 0000 ff01 d82b xxxx xxxx
                         yyyy yyyy 0800 f015 7a3c 0000 5dc5 0d3a
                         17ae 0b00

00:17:02.064284 ppp0 < y.y.y.y > x.x.x.x: icmp: echo reply (ttl 111, id
29961)
                         4500 0024 7509 0000 6f01 2696 yyyy yyyy
                         xxxx xxxx 0000 f815 7a3c 0000 5dc5 0d3a
                         17ae 0b00
```

Another OS that behaves the same is ULTRIX:

```
[root@godfather precedence_echo]# /usr/local/bin/sing -c 2 -TOS 1
y.y.y.y
SINGing to y.y.y.y (y.y.y.y): 16 data bytes
16 bytes from y.y.y.y: seq=0 ttl=237 TOS=0 time=371.776 ms

--- y.y.y.y sing statistics ---
2 packets transmitted, 1 packets received, 50% packet loss
```

```
round-trip min/avg/max = 371.776/371.776/371.776 ms
[root@godfather precedence_echo]#
```

We will use, again, the IP TTL field value to differentiate between the two operating systems.

6.7.1 Changed Pattern with Replies for Different ICMP Query Types

We have a changed pattern with Microsoft Windows 98/98SE/ME when using other ICMP Query message types other than ICMP Echo Request. Instead of echoing this field back, they will zero out this field with their replies.

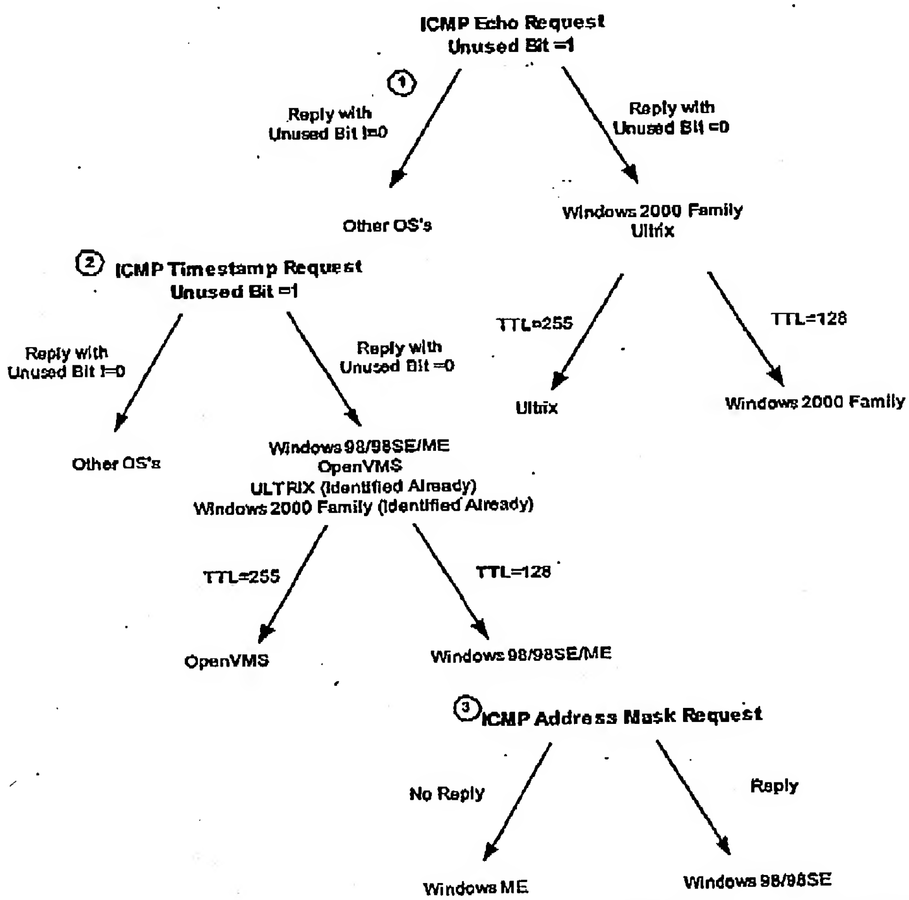


Diagram 8: An example for a way to fingerprint operating systems using the unused bit in the TOS Byte echoing method

Further distinction between the Microsoft operating systems can be achieved if we will query them with ICMP Address Mask request, which only Microsoft Windows 98/98SE will answer for. The Microsoft Windows ME will not reply, enabling us to identify it.

Operating System	Information Request With Unused=1	Time Stamp Request With Unused=1	Address Mask Request With Unused=1	Echo Request With Unused=1
Debian GNU/ LINUX 2.2, Kernel 2.4 test 2	Not Answering	0x1	Not Answering	0x1
Redhat LINUX 6.2 Kernel 2.2.14	Not Answering	0x1	Not Answering	0x1
FreeBSD 4.0	Not Answering	0x1	Not Answering	0x1
FreeBSD 4.1.1	Not Answering.	0x1	Not Answering	0x1
OpenBSD 2.7	Not Answering		Not Answering	
OpenBSD 2.6	Not Answering		Not Answering	
NetBSD	Not Answering		Not Answering	
BSDI BSD/OS 4.0	Not Answering		Not Answering	
BSDI BSD/OS 3.1	Not Answering		Not Answering	
Solaris 2.5.1	Not Implemented			
Solaris 2.6	Not Implemented	0x1	0x1	0x1
Solaris 2.7	Not Implemented	0x1	0x1	0x1
Solaris 2.8	Not Implemented	0x1	0x1	0x1
HP-UX v10.20			Not Answering	
HP-UX v11.0	Not Answering	Not Answering	0x1	0x1
Compaq Tru64 v5.0	0x1	0x1	Not Answering	0x1
AIX 4.3	0x1	0x1	Not Answering	0x1
AIX 4.2.1	0x1	0x1	Not Answering	0x1
AIX 4.1	0x1	0x1	Not Answering	0x1
AIX 3.2	0x1	0x1	Not Answering	0x1
ULTRIX 4.2 – 4.5	0x0	0x0	0x0	0x0
OpenVMS v7.1-2	0x1	0x1	0x1	0x1
Windows 95	Not Answering	Not Answering		
Windows 98	Not Answering	0x0	0x0	0x1
Windows 98 SE	Not Answering	0x0	0x0	0x1
Windows ME	Not Answering	0x0	Not Answering	0x1
Windows NT 4 WRKS SP 3	Not Answering	Not Answering		
Windows NT 4 WRKS SP 6a	Not Answering	Not Answering	Not Answering	0x1
Windows NT 4 Server SP4	Not Answering	Not Answering	Not Answering	
Windows 2000 Professional	Not Answering	0x0	Not Answering	0x0
Windows 2000 Server	Not Answering	0x0	Not Answering	0x0

Table 12: ICMP Query Message Types with the TOS Byte Unused Bit value ! = 0

6.8 Using the Unused (Identifying Sun Solaris & HP-UX 10.30 & 11.0x OS based machines)

RFC 791 defines a three bits field used for various control flags in the IP Header. Bit 0 of this bits field is the reserved flag, and must be zero according to the RFC.

What will happen if we will decide to break this definition and send our ICMP Query requests with this bit set (having the value of one)?

Sun Solaris & HPUX 11.0x (possibly 10.30 as well) will echo back the reserved bit.

This is a tcpdump trace describing an ICMP Echo request sent with the reserved Bit set, and the ICMP Echo reply we received echoing the reserved bit. This trace was produced against an HP-UX 11.0 machine:

```
21:31:21.033366 if 4  > y.y.y.y > x.x.x.x: icmp: echo request (ttl 255,
id 13170)
                   4500 0024 3372 8000 ff01 fc8c yyyy yyyy
                   xxxx xxxx 0800 8b1b 8603 0000 f924 bd39
                   3082 0000
21:31:21.317916 if 4  < x.x.x.x > y.y.y.y: icmp: echo reply (ttl 236,
id 25606)
                   4500 0024 6406 8000 ec01 def8 xxxx xxxx
                   yyyy yyyy 0000 931b 8603 0000 f924 bd39
                   3082 0000
```

The next trace was produced against a Sun Solaris 2.8 machine:

```
16:51:37.470995 if 4  > 195.72.167.220 > x.x.x.x: icmp: echo request
(ttl 255, id 13170)
                   4500 0024 3372 8000 ff01 e0e1 c348 a7dc
                   xxxx xxxx 0800 edae 3004 0000 69e3 bc39
                   ad2f 0700
16:51:37.745254 if 4  < x.x.x.x > 195.72.167.220: icmp: echo reply (DF)
(ttl 243, id 5485)
                   4500 0024 156d c000 f301 cae6 xxxx xxxx
                   c348 a7dc 0000 f5ae 3004 0000 69e3 bc39
                   ad2f 0700
```

If we examine these traces closely we can identify a distinction between them. The DF bit is set with the Sun Solaris ICMP Query reply and not with the HP-UX 11.0 OS reply.

If you recall from the "DF Bit Playground" section Sun Solaris would set the DF bit by default with all its ICMP Query replies. HP-UX 10.30, and 11.0x operating systems would initiate, by default, a proprietary method in order to determine the PMTU using ICMP Echo requests. After the PMTU is determined the following ICMP Query replies would have the DF bit set in the IP Header.

If we are using only one datagram than in most cases we can distinguish between Sun Solaris and HP-UX 10.30, and 11.0x operating systems since the DF bit will not be set (and if the PMTU is not already determine).

All ICMP Query replies on the same operating system use the same pattern (either echo the reserved bit with all replies or not). This enable us to use another ICMP Query message type for this fingerprinting method. If we send an ICMP Address Mask request with the reserved bit set, the result a Sun Solaris 2.8 machine will produce will be something like the next trace:

```
18:39:32.262869 if 4  > y.y.y.y > x.x.x.x : icmp: address mask request
(ttl 255, id 13170)
                   4500 0020 3372 8000 ff01 e12e yyyy yyyy
                   xxxx xxxx 1100 a0fb 4e04 0000 0000 0000
18:39:32.561373 if 4  < x.x.x.x > y.y.y.y: icmp: address mask is
0xffffff00 (DF) (ttl 243, id 51792)
                   4500 0020 ca50 c000 f301 1650 xxxx xxxx
                   yyyy yyyy 1200 a0fa 4e04 0000 ffff ff00
```

We will have both the reserved bit and the DF bit set on the ICMP Address Mask reply, a unique pattern Sun Solaris machines have with ICMP Query replies.

This operating system fingerprinting method enables us to identify and distinguish between Sun Solaris, and HP-UX 10.30 &11.0x operating systems to the other operating systems.

I have asked Alfredo Andres Omella, author of SING, to incorporate the ability to set the reserved bit with his tool. Alfredo has introduced the –U option along with the ability to identify if this bit is set on the reply (if any) we get, with the latest version of SING:

```
[root@godfather bin]# ./sing -mask -U IP_Address
SINGing to IP_Address (IP_Address): 12 data bytes
12 bytes from IP_Address: icmp_seq=0 RF! DF! ttl=243 TOS=0 mask=255.255.255.0
12 bytes from IP_Address: icmp_seq=1 RF! DF! ttl=243 TOS=0 mask=255.255.255.0
12 bytes from IP_Address: icmp_seq=2 RF! DF! ttl=243 TOS=0 mask=255.255.255.0
12 bytes from IP_Address: icmp_seq=3 RF! DF! ttl=243 TOS=0 mask=255.255.255.0
12 bytes from IP_Address: icmp_seq=4 RF! DF! ttl=243 TOS=0 mask=255.255.255.0
— IP_Address sing statistics —
5 packets transmitted, 5 packets received, 0% packet loss
[root@godfather bin]#
```

This method was tested against: Linux Kernel 2.4 test 2,4,5,6; Linux Kernel 2.2.x; FreeBSD 4.0, 3.4; OpenBSD 2.7,2.6; NetBSD 1.4.1,1.4.2; BSDI BSD/OS 4.0,3.1; Solaris 2.6,2.7,2.8; HP-UX 10.20, 11.0; Compaq Tru64 5.0; Aix 4.1,3.2; Irix 6.5.3, 6.5.8; Ultrix 4.2 – 4.5; OpenVMS v7.1-2; Novel Netware 5.1 SP1, 5.0, 3.12; Microsoft Windows 98/98SE, Microsoft Windows NT WRKS SP6a, Microsoft Windows NT Server SP4, Microsoft Windows 2000 Family.

6.9 DF Bit Echoing

Some operating systems, when receiving an ICMP Query message with the DF bit set, would set the DF bit with their replies as well. Sometimes it would be in contrast with their regular behavior, which would be not setting the DF Bit in their replies for a regular query that comes with the DF bit not set.

6.9.1 DF Bit Echoing with the ICMP Echo request

The tcpdump trace below illustrates an ICMP Echo request sent from a Linux box, using SING[42], to a SUN Solaris 2.7 machine:

```
[root@godfather bin]# ./sing -echo -G IP_Address
SINGing to IP_Address (IP_Address): 16 data bytes
16 bytes from IP_Address: icmp_seq=0 DF! ttl=243 TOS=0 time=188.289 ms
16 bytes from IP_Address: icmp_seq=1 DF! ttl=243 TOS=0 time=250.026 ms
16 bytes from IP_Address: icmp_seq=2 DF! ttl=243 TOS=0 time=240.298 ms
16 bytes from IP_Address: icmp_seq=3 DF! ttl=243 TOS=0 time=260.036 ms

--- IP_Address sing statistics ---
4 packets transmitted, 4 packets received, 0% packet loss
round-trip min/avg/max = 188.289/234.662/260.036 ms
[root@godfather bin]#
```

The tcpdump trace:

```
17:16:23.527050 if 4  > x.x.x.x > y.y.y.y: icmp: echo request (DF) (ttl
255, id 13170)
                         4500 0024 3372 4000 ff01 20f0 xxxx xxxx
```

[42] The –G option with SING sets the DF Bit with the ICMP Query requests.

```
                     yyyy yyyy 0800 a5cd b304 0000 37e9 bc39
                     a30a 0800
17:16:23.715250 if 4  < y.y.y.y > x.x.x.x: icmp: echo reply (DF) (ttl
243, id 18227)
                     4500 0024 4733 4000 f301 192f yyyy yyyy
                     xxxx xxxx 0000 adcd b304 0000 37e9 bc39
                     a30a 0800
```

Most of the operating systems that I have checked this behavior against did the same thing. In the reply they produced, the DF bit was set.

Which operating systems are the exceptional and do not echo back the DF bit?
Linux operating systems based on Kernel 2.2.x, and Kernel 2.4 with the various test kernels, Ultrix v4.2 – 4.5, and Novell Netware.

How can we distinguish between those operating systems?
Frankly it is quite simple. Since LINUX and Ultrix are using an IP TTL field value of 255 in their ICMP Query replies, and Novell Netware uses 128, it is easy to distinguish between those groups. If we want to further distinguish between LINUX based systems and Ultrix based systems, we can send an ICMP Information request or an ICMP Address Mask request to the questioned machines. The machine, which would answer those, will be the one based on the Ultrix operating system.

6.9.2 DF Bit Echoing with the ICMP Address Mask request

With ICMP Address Mask requests we have a different story. Among the operating systems that I have checked that answer for an ICMP Address Mask request Sun Solaris & OpenVMS echo back the DF bit. Microsoft Windows 98, Microsoft Windows 98 SE, and Ultrix do not echo back the DF bit.

Again it is very simple to distinguish between the Microsoft Windows 98 family and between the Ultrix machines. Since the Microsoft Windows 98 family is using 128 as their IP TTL field value in their ICMP query replies and Ultrix uses 255, we can distinguish between those operating systems.

We have here a simple method to distinguish between Microsoft Windows 98 / 98 SE, and Ultrix machines to the rest of the operating systems world.

Another interesting piece of information is that the Microsoft Windows 98 family changed its behavior from DF echoing with the ICMP Echo request to not echoing with the ICMP Address Mask request. This inconsistency is a factor with all Microsoft operating systems (Echoing with ICMP Echo request, not echoing with the other types of ICMP query).

6.9.3 DF Bit Echoing with the ICMP Timestamp request

Since a lot more operating systems answer for an ICMP Timestamp request than with the ICMP Address Mask request, we have a bit more difficulty in identifying those.

Linux machines based on Kernel 2.2.x, or Kernel 2.4, Ultrix, Microsoft Windows 98/98SE/ME, and the Microsoft Windows 2000 Family would not echo back the DF bit with ICMP Timestamp replies they produce for ICMP Timestamp request that sets their DF bit.

Here we can only distinguish between certain groups of operating systems; again it would be according to their IP TTL field value with their replies.

Linux would use 255 as its TTL field value for the ICMP Timestamp reply; Ultrix would use the same value. The Microsoft family of operating systems that would answer for this kind of query would use 128 as their TTL value.

Again we have Linux and Ultrix on the one hand and the Microsoft Family on the other hand. How can we further distinguish between those? We can correlate the information (as discussed in the next section, or query the "suspicious" machines with another type of ICMP Query message).

6.9.4 Using all of the Information in order to identify maximum of operating systems

We can group Linux and Ultrix with the ICMP Echo requests. We can do the same with Microsoft Windows 98 / 98 SE & Ultrix using the ICMP Address Mask requests. This would allow us to pinpoint the Linux boxes from the first stage. So when we would go into the third stage we would know which operating systems are Linux based, which are Microsoft Windows 98 / 98 SE based, and which are Ultrix based. This would leave us with Microsoft Windows ME and with the Microsoft Windows 2000 family machines.

6.9.5 Why this would work (for the skeptical)

All those skeptical would say that if they receive an ICMP Query request with the DF bit set than it should be clear that something is wrong and someone is probably trying to scan them. Think again. What would happen if a Sun Solaris machine will query your machine? Than the same behavior would be produced.

This is an ICMP Echo request sent from a Solaris 2.6 box to a Linux box. We can see that the DF bit is set with the request and not set with the reply. But again if some one would mimic this behavior with a tool used on a Linux box to query the world, which is 100% mimicking a Sun Solaris request than we would never know if this is a legit request or an attempt for scanning / fingerprinting.

```
Initializing Network Interface...
Decoding raw data on interface ppp0

-*> Snort! <*-
Version 1.6
By Martin Roesch (roesch@clark.net, www.clark.net/-roesch)
08/10-23:32:52.201612 y.y.y.y -> 139.92.207.58
ICMP TTL:239 TOS:0x0 ID:48656  DF
ID:2080    Seq:0  ECHO
39 93 10 A3 00 03 F0 E5 08 09 0A 0B 0C 0D 0E 0F   9...............
10 11 12 13 14 15 16 17 18 19 1A 1B 1C 1D 1E 1F   ................
20 21 22 23 24 25 26 27 28 29 2A 2B 2C 2D 2E 2F    !"#$%&'()*+,-./
30 31 32 33 34 35 36 37                           01234567

08/10-23:32:52.201649 139.92.207.58 -> y.y.y.y
ICMP TTL:255 TOS:0x0 ID:349
ID:2080    Seq:0  ECHO REPLY
39 93 10 A3 00 03 F0 E5 08 09 0A 0B 0C 0D 0E 0F   9...............
10 11 12 13 14 15 16 17 18 19 1A 1B 1C 1D 1E 1F   ................
20 21 22 23 24 25 26 27 28 29 2A 2B 2C 2D 2E 2F    !"#$%&'()*+,-./
30 31 32 33 34 35 36 37                           01234567
```

Operating System	Info. Request	Time Stamp Request	Address Mask Request	Echo Request
Debian GNU/ LINUX 2.2, Kernel 2.4 test 2	Not Answering	+ (- DF)	Not Answering	+ (- DF)
Redhat LINUX 6.2 Kernel 2.2.14	Not Answering	+ (- DF)	Not Answering	+ (- DF)
FreeBSD 4.0	Not Answering	+ (+ DF)	Not Answering	+ (+ DF)
FreeBSD 3.4	Not Answering	+ (+ DF)	Not Answering	+ (+ DF)
OpenBSD 2.7	Not Answering	+ (+ DF)	Not Answering	+ (+ DF)
OpenBSD 2.6	Not Answering	+ (+ DF)	Not Answering	+ (+ DF)
NetBSD	Not Answering	+ (+ DF)	Not Answering	+ (+ DF)
BSDI BSD/OS 4.0	Not Answering	+ (+ DF)	Not Answering	+ (+ DF)
BSDI BSD/OS 3.1	Not Answering	+ (+ DF)	Not Answering	+ (+ DF)
Solaris 2.5.1	Not Answering	+ (+ DF)	+ (+ DF)	+ (+ DF)
Solaris 2.6	Not Answering	+ (+ DF)	+ (+ DF)	+ (+ DF)
Solaris 2.7	Not Answering	+ (+ DF)	+ (+ DF)	+ (+ DF)
Solaris 2.8	Not Answering			
HP-UX v10.20	Not Answering	Not Answering	Not Answering	+ (+ DF)
HP-UX v11.0			+ (+ DF)	
Compaq Tru64 v5.0		+ (+ DF)	Not Answering -	+ (+ DF)
Irix 6.5.3	Not Answering	+ (+ DF)	Not Answering	+ (+ DF)
Irix 6.5.8	Not Answering	+ (+ DF)	Not Answering	+ (+ DF)
AIX 4.1		+ (+ DF)	Not Answering	+ (+ DF)
AIX 3.2		+ (+ DF)	Not Answering	+ (+ DF)
ULTRIX 4.2 – 4.5		+ (- DF)	+ (- DF)	+ (- DF)
OpenVMS v7.1-2		+ (+ DF)	+ (+ DF)	+ (+ DF)
Novell Netware 5.1 SP1	Not Answering	Not Answering	Not Answering	+ (- DF)
Novell Netware 5.0	Not Answering	Not Answering	Not Answering	+ (- DF)
Novell Netware 3.12	Not Answering	Not Answering	Not Answering	+ (- DF)
Windows 95	Not Answering	Not Answering	+ (- DF)	+ (+ DF)
Windows 98	Not Answering	+ (- DF)	+ (- DF)	+ (+ DF)
Windows 98 SE	Not Answering	+ (- DF)	Not Answering	+ (+ DF)
Windows ME	Not Answering	+ (- DF)		
Windows NT 4 WRKS SP 3	Not Answering	Not Answering	Not Answering	+ (+ DF)
Windows NT 4 WRKS SP 6a	Not Answering	Not Answering	Not Answering	+ (+ DF)
Windows NT 4 Server SP4	Not Answering	Not Answering	Not Answering	+ (+ DF)
Windows 2000 Professional	Not Answering	+ (- DF)	Not Answering	+ (+ DF)
Windows 2000 Server	Not Answering	+ (- DF)		

Table 13: DF Bit Echoing

6.9.6 Combining all together

If we combine every thing together than we can start from sending ICMP Echo requests with the DF bit set probing the targeted host(s) / network(s). The operating systems, which will not echo the DF bit with their ICMP Query replies, will be Linux operating system machines based either on kernel 2.2.x, or on Kernel 2.4.x, Novell Netware, and Ultrix machines. We can distinguish between the Novell Netware machines, the Linux based machines, and the Ultrix based machines according to the IP TTL field values with the ICMP Echo replies.

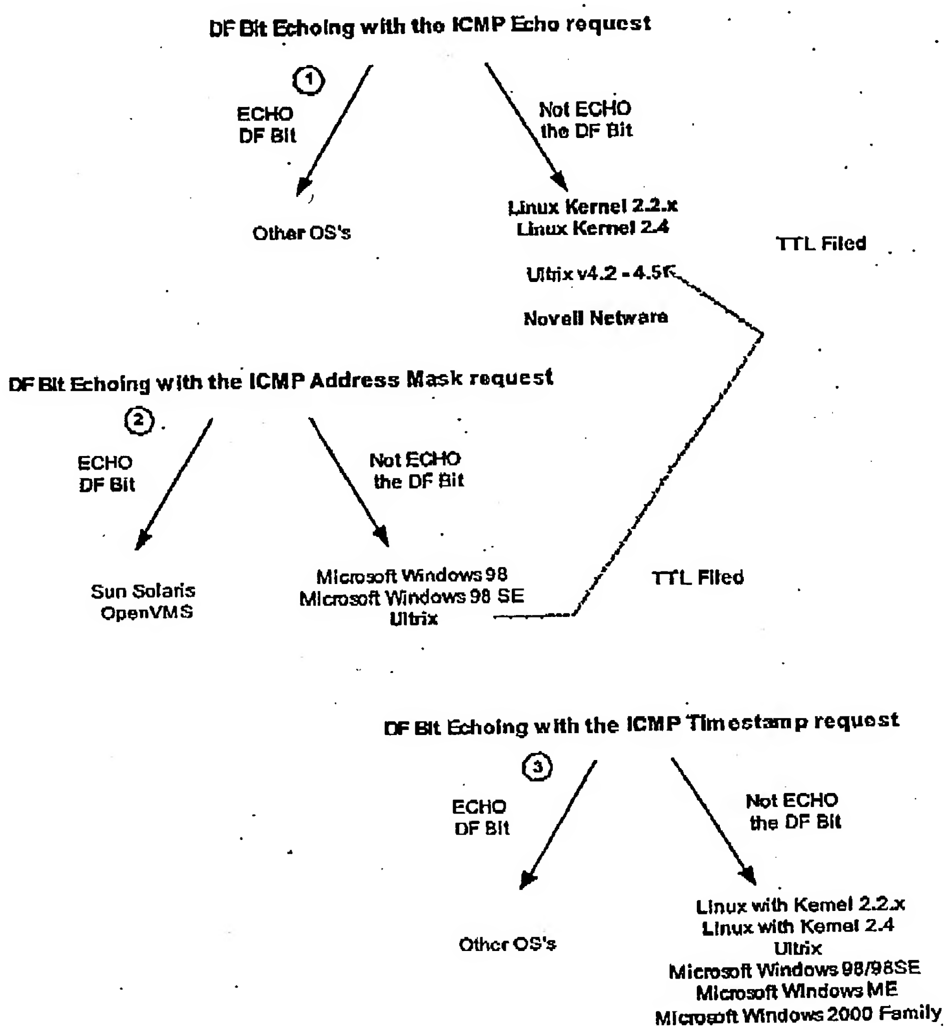


Diagram 9: An example of fingerprinting using the DF Bit Echoing technique

The second stage would be sending ICMP Address Mask requests with the DF bit set to the same-targeted host(s). Microsoft Windows 98/98 SE and Ultrix based machines would not echo the DF bit with their ICMP Query reply. We than can distinguish between the Ultrix machines and the Microsoft Windows machines, because of the different IP TTL field values in the ICMP Address Mask replies.

We can now also identify the Ultrix machines from the first step – we know their IPs now. Than it leaves us with only the Linux boxes. Within two steps we are able of fingerprinting Novell Netware, Ultrix, Microsoft Windows 98/98 SE and Linux operating systems based on kernel 2.2.x or on kernel 2.4.x.

In the last step of this example we are sending ICMP Timestamp requests with the DF bit set to the same group of IPs we are probing. The operating systems which do not echo back the DF bit in their ICMP Query replies are Linux operating systems based on Kernel 2.2.x, or on Kernel 2.4, Ultrix, Microsoft Windows 98/98SE, Microsoft Windows ME, and Microsoft Windows 2000 Family. Since we already fingerprinted most of the operating systems in this it enable us to fingerprint Microsoft Windows ME, and Microsoft Windows 2000 family based machines.

6.10 Using Code field values different than zero within ICMP ECHO requests

An interesting detail I have discovered during the lab experiments I did when I have researched ICMP scanning is when a wrong code is sent along with the correct type of ICMP query message, different operating systems would send different code values back.

In the next example I have sent an ICMP Echo Request with the code field value set to 38 instead of 0, to a LINUX machine running Redhat LINUX 6.2 Kernel 2.2.14.

We can look at the tcpdump trace, the type and code fields are in bold type:

```
00:21:05.238649 ppp0 > x.x.x.x > y.y.y.y: icmp: echo request (ttl 255,
id 13170)
                         4500 0024 3372 0000 ff01 08d3 xxxx xxxx
                         yyyy yyyy 0826 af13 2904 0000 41e4 c339
                         17a4 0300
00:21:05.485617 ppp0 < y.y.y.y > x.x.x.x: icmp: echo reply (ttl 240, id
2322)
                         4500 0024 0912 0000 f001 4233 yyyy yyyy
                         xxxx xxxx 0026 b713 2904 0000 41e4 c339
                         17a4 0300
```

In the ICMP Echo reply LINUX produced the code field value is set to 38.

If we examine what RFC 792 requires, we see that LINUX does exactly that.

The sending side initializes the identifier (used to identify ECHO requests aimed at different destination hosts) and sequence number (if multiple ECHO requests are sent to the same destination host), adds some data (arbitrary) to the data field and sends the ICMP ECHO Request to the destination host. *In the ICMP header the code equals zero.* The recipient should *only change* the type to ECHO Reply and return the datagram to the sender.

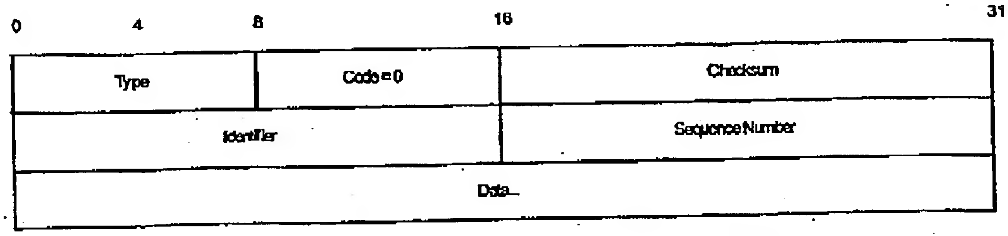


Figure 12: ICMP ECHO Request & Reply message format

This also means that we trust another machine to behave correctly, when that host produce the ICMP Echo reply.

LINUX changes the type field value to 0 and sends the reply. The code field is unchanged.

I have checked the behavior of my Microsoft Windows 2000 Professional box. I have sent the same ICMP ECHO Request message to the Microsoft Windows box (the code field is in bold type):

```
10:03:33.860212 eth0 > localhost.localdomain > 192.168.1.1: icmp: echo
request
                         4500 0020 3372 0000 fe01 0614 c0a8 0105
                         c0a8 0101 0826 d618 6102 f658 0183 c8e2

10:03:33.860689 eth0 < 192.168.1.1 > localhost.localdomain: icmp: echo
reply
                         4500 0020 2010 0000 8001 9776 c0a8 0101
                         c0a8 0105 0000 de3e 6102 f658 0183 c8e2
                         0000 0000 0000 0000 0000 0000 0000
```

The Microsoft Windows 2000 Professional operating system changed the code field value on the ICMP Echo Reply to the value of 0.

This method was tested with various operating systems including LINUX Kernel 2.4t1-8, IBM AIX 4.x & 3.2, SUN Solaris 2.51, 2.6, 2.7 & 2.8, OpenBSD 2.6 & 2.7, NetBSD 1.4.1, 1.4.2, BSDI BSD/OS 4.0 & 3.1, HP-UX 10.20 & 11.0, Compaq Tru64 v5.0, Irix 6.5.3 & 6.5.8, Ultrix 4.2-4.5, OpenVMS, FreeBSD 3.4, 4.0 & 4.1 and they produced the same results as the LINUX box (Kernel 2.2.x) did.

Microsoft Windows 4.0 Server SP4, Microsoft Windows NT 4.0 Workstation SP 6a, Microsoft Windows NT 4.0 Workstation SP3, Microsoft Windows 95 / 98 / 98 SE / ME have produced the same behavior as the Microsoft Windows 2000 Professional (Server & Advanced Server).

We have a fingerprinting method to differentiate between a Microsoft Windows based machine to the rest of the operating systems world using code values, which are different than zero, inside ICMP Echo Requests.

6.11 Using Code field values different than zero within ICMP Timestamp Request

I have decided to map which operating systems would answer to an ICMP Timestamp Request that would have its code field not set to zero, and how the ICMP Timestamp reply (if any) will help us identify those operating systems.

6.11.1 The non-answering Operating Systems

Interesting results were produced. The Microsoft Windows 98/98 SE/ME, and the Microsoft Windows 2000 Family that have answered to ICMP Timestamp requests with the code field set to zero, now did not produce any reply back.

This enables us to group together certain versions of the Windows Operating System.

The next diagram shows how we can distinguish between the different Microsoft Windows operating systems using two datagrams of ICMP Timestamp request. The first one is a regular one; the Microsoft Windows machines that do not answer are Microsoft Windows 95 and Microsoft Windows NT 4.0 Workstation with SP 6a. All other operating systems (that I have tested) answer the ICMP Time stamp request. The second stage is sending another datagram, this time with the Code field set to a value, which is not equal to zero. The operating systems that would not answer will include Windows 98/98 SE/ME/2000 Family, which are the newer versions of Microsoft Windows operating systems.

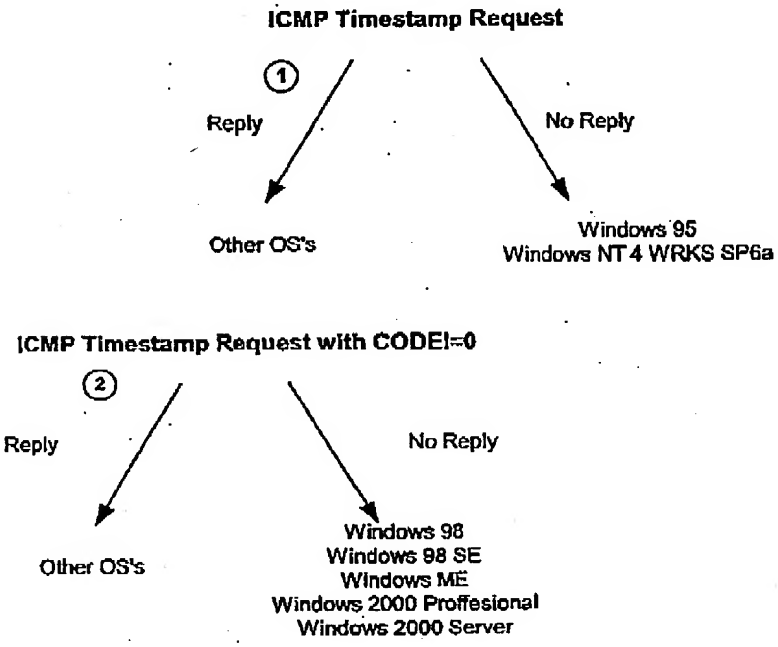


Diagram 10: Finger Printing Using ICMP Timestamp Request and Wrong Codes

It is quite obvious that Microsoft have tried to change some of their newer operating systems fingerprinting in later TCP/IP implementations of their operating systems. For example, the default for answering an ICMP Timestamp request was changed from "no answer" to "answer", like UNIX and UNIX-like operating systems. But the Microsoft programmers / designers / architects / security engineers did not think about every thing apparently.

6.11.2 Operating Systems the Zero out the Code field value on Reply

I was looking to see if there are any operating systems in which answered the crafted ICMP Timestamp Query with the Code field set to a value different than zero, which might zero out this field value in its ICMP Echo Reply.

I have found that LINUX operating systems based on Kernel 2.2.x or on the 2.4 Kernel (with the various test Kernels) zero out the code field with the ICMP Echo replies they produce. The next trace is a tcpdump trace describing ICMP Echo Request and reply from a LINUX 2.4 test Kernel 6, to a crafted ICMP Echo Request with a code field different than zero:

```
[root@godfather /root]# sing  -tstamp -x 38 -c 2 IP_Address
SINGing to IP_Address (IP_Address): 20 data bytes
20 bytes from IP_Address: icmp_seq=0 ttl=243 TOS=0 diff=24315927
20 bytes from IP_Address: icmp_seq=1 ttl=243 TOS=0 diff=24316176

--- IP_Address sing statistics ---
2 packets transmitted, 2 packets received, 0% packet loss
[root@godfather /root]#
```

```
20:10:18.138486 ppp0 > x.x.x.x > y.y.y.y: icmp: time stamp request (ttl
255, id 13170)
                         4500 0028 3372 0000 ff01 606c xxxx xxxx
                         yyyy yyyy 0d26 2e0c 7c04 0000 03af 451a
                         0000 0000 0000 0000
20:10:18.354222 ppp0 < y.y.y.y > x.x.x.x: icmp: time stamp reply (ttl
243, id 15717)
                         4500 0028 3d65 0000 f301 6279 yyyy yyyy
                         xxxx xxxx 0e00 888b 7c04 0000 03af 451a
                         0422 4e31 0422 4e31
20:10:19.134165 ppp0 > x.x.x.x > y.y.y.y: icmp: time stamp request (ttl
255, id 13170)
                         4500 0028 3372 0000 ff01 606c xxxx xxxx
                         yyyy yyyy 0d26 2928 7c04 0100 03af 48fe
                         0000 0000 0000 0000
20:10:19.354210 ppp0 < y.y.y.y > x.x.x.x: icmp: time stamp reply (ttl
243, id 15718)
                         4500 0028 3d66 0000 f301 6278 yyyy yyyy
                         xxxx xxxx 0e00 7bed 7c04 0100 03af 48fe
                         0422 520e 0422 520e
```

6.11.3 Changed Patterns

The LINUX operating system behavior with the crafted ICMP Timestamp requests is in contrast with its behavior with the crafted ICMP Echo Requests sent with the Code field set to a value different than zero.

This also gives us a unique piece of information that enables us to identify LINUX machines better.

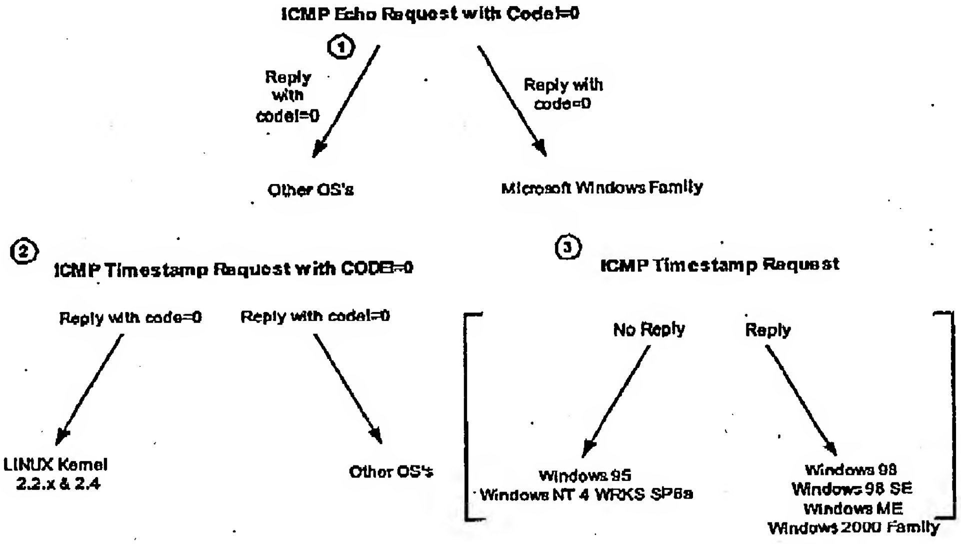


Diagram 11: An Example of Finger Printing Using crafted ICMP Echo & Timestamp Request

The diagram above describes a process in which we can use in order to differentiate between certain groups of operating systems.

The first step is sending an ICMP Echo request with the code field set to a value different than zero. The ICMP Echo replies with the code field equal to zero would distinguish the Microsoft based operating systems group, from the other UNIX and UNIX-like operating systems.

Sending ICMP Timestamp requests with the Code field value different than zero to all participants of the group of the UNIX and UNIX-like operating systems will identify LINUX 2.2.x and 2.4 Kernel based machines (since they zero out the code field in their replies).

Sending ICMP Timestamp request to the Microsoft Windows based group of operating systems will separate the group to those machines rather being windows 95 or windows NT 4 SP4 and above (not answer the query), to those that may be one of the following – Microsoft Windows 98 / SE / ME / Windows 2000 Family (answer the query).

For a list of ICMP Query message types sent to different types of operating systems with the code field set to a value different than zero, and the various ICMP Query replies we got back (if any) please see "Appendix D: ICMP Query Message types with Code field !=0 (table)".

Using the ICMP Error Messages

6.12 Operating system, which do not generate ICMP Protocol Unreachable Error Messages

Several operating systems will not generate an ICMP Protocol Unreachable error message, when one is expected to be produced, in response to an offending datagram trying to use a protocol, which is not used on those operating systems.

Those operating systems include:

- AIX
- DG-UX
- HP-UX

6.13 ICMP Error Message Quenching

RFC 1812 and RFC 1122 suggest limiting the rate at which various error messages are sent. Only few operating systems are known to follow this.

An attacker can use this to send UDP packets to a random, high UDP port and count the number of ICMP Destination unreachable messages received within a given amount of time.

6.14 ICMP Error Message Quoting Size

Each ICMP error message includes the Internet Protocol (IP) Header and *at least* the first 8 data bytes of the datagram that triggered the error (the offending datagram); more than 8 bytes *may* be sent according to RFC 1122.

Most of the operating systems will quote the offending packets IP Header and the first 8 data bytes of the datagram that triggered the error. Several operating systems and networking devices will parse the RFC guidelines a bit different and will echo more than 8 bytes.

Which operating systems will quote more?
LINUX based on Kernel 2.0.x/2.2.x/2.4.t-x, Sun Solaris, HPUX 11.x, MacOS 7.55/8.x/9.04, Nokia boxes, Foundry Switches (and other OSs and several Networking Devices) are a good example.

The fact is not new. Fyodor outlined this in his article "Remote OS Identification by TCP/IP Fingerprinting"[43].

The idea is in trying to differentiate between the different operating systems that quote more than the usual. How can this be done? Looking for example on the amount of information quoted. Is there a limit to the quoted size? Will the quoted data be the entire offending packet or just part of it? Will the quoted data be the echoed correctly? Will extra bytes will be padded to the echoed data? and some other parameters.

The next example is with Sun Solaris 7. I have sent a UDP datagram to a closed UDP port:

```
00:13:35.559947 ppp0 > x.x.x.x.1084 > y.y.y.y.2000: udp 0 (ttl 64, id
44551)
                         4500 001c ae07 0000 4011 7aa4 xxxx xxxx
                         yyyy yyyy 043c 07d0 0008 a1ac

00:13:35.923691 ppp0 < y.y.y.y > x.x.x.x: icmp: y.y.y.y udp port 2000
unreachable Offending pkt: x.x.x.x.1084 > y.y.y.y.2000: udp 0 (ttl 45,
id 44551) (DF) (ttl 236, id 63417)
                         4500 0038 f7b9 4000 ec01 44e5 yyyy yyyy
                         xxxx xxxx 0303 4f3c 0000 0000 4500 001c
                         ae07 0000 2d11 8da4 xxxx xxxx yyyy yyyy
```

[43] http://www.insecure.org/nmap/nmap-fingerprinting-article.html

```
                    043c 07d0 0008 a1ac
```

Please note that for having more than 8 data bytes quoted, you need to have data in the offending datagram. If not, there is nothing to quote beyond the regular 8 bytes (usually, if the OS is not padding other data bytes).

The next example is with Sun Solaris 8. I have sent a UDP datagram to a closed UDP port, adding 80 bytes of data to the datagram.

```
[root@godfather]# hping2 -2 -d 80 -c 1 y.y.y.y
eth0 default routing interface selected (according to /proc)
HPING y.y.y.y (eth0 y.y.y.y): udp mode set, 28 headers + 80 data bytes
ICMP Port Unreachable from y.y.y.y (y.y.y.y)

--- y.y.y.y hping statistic ---
1 packets tramitted, 0 packets received, 100% packet loss
round-trip min/avg/max = 0.0/0.0/0.0 ms
```

The tcpdump trace:

```
11:52:50.830383 eth0 > x.x.x.x.2198 > y.y.y.y.0: udp 80 (ttl 64, id
17240)
                    4500 006c 4358 0000 4011 99ae xxxx xxxx
                    yyyy yyyy 0896 0000 0058 8b5f 5858 5858
                    5858 5858 5858 5858 5858 5858 5858 5858
                    5858 5858 5858 5858 5858 5858 5858 5858
                    5858 5858 5858 5858 5858 5858 5858 5858
                    5858 5858 5858 5858 5858 5858 5858 5858
                    5858 5858 5858 5858 5858 5858

11:52:51.367331 eth0 < y.y.y.y > x.x.x.x: icmp: y.y.y.y udp port 0
unreachable Offending pkt: x.x.x.x.2198 > y.y.y.y.0: udp 80 (ttl 48, id
17240) (DF) (ttl 231, id 49576)
                    4500 0070 c1a8 4000 e701 3469 yyyy yyyy
                    xxxx xxxx 0303 bf05 0000 0000 4500 006c
                    4358 0000 3011 a9ae xxxx xxxx yyyy yyyy
                    0896 0000 0058 8b5f 5858 5858 5858 5858
                    5858 5858 5858 5858 5858 5858 5858 5858
                    5858 5858 5858 5858 5858 5858 5858 5858
                    5858 5858 5858 5858 5858 5858 5858 5858
```

The result is an ICMP Port Unreachable Error message that will echo only 64 bytes of the offending datagram's data portion.

The limit of 64 bytes quoted from the offending packet's data portion is not limited to Sun Solaris only. HPUX 11.x, MacOS 7.55/8.x/9.04, will do the same.

Other operating systems / networking devices will have their own barriers. For example, LINUX based on Kernel 2.2.x/2.4.x-t will send and ICMP Error Message up to 576 bytes long. LINUX will quote 528 bytes from the data portion of the offending packet (576 minus 20 bytes of usuall IP Header, minus 8 bytes of the ICMP Header, minus the offending packet's IP Header that is 20 bytes will leave you with 528 bytes of data portion. This is no IP options are presented).

I know an operating system, and a family of networking devices that will pad extra data to the echoed offending packet. LINUX case is detailed in the next section. The next example is with

Foundry Networks ServerIron running software version 07.1.02T12. I have sent a UDP datagram to a closed UDP port on the Foundry switch:

```
[root@godfather]# hping2 -2 -c 1 y.y.y.y
eth0 default routing interface selected (according to /proc)
HPING y.y.y.y (eth0 y.y.y.y): udp mode set, 28 headers + 0 data bytes
ICMP Port Unreachable from y.y.y.y (y.y.y.y)

--- y.y.y.y hping statistic ---
1 packets tramitted, 0 packets received, 100% packet loss
round-trip min/avg/max = 0.0/0.0/0.0 ms
[root@godfather]#


12:08:47.793503 eth0 > x.x.x.x.2498 > y.y.y.y.0: udp 0 (ttl 64, id
44437)
                         4500 001c ad95 0000 4011 885f xxxx xxxx
                         yyyy yyyy 09c2 0000 0008 b13f

12:08:48.240208 eth0 < y.y.y.y > x.x.x.x: icmp: y.y.y.y udp port 0
unreachable Offending pkt: x.x.x.x.2498 > y.y.y.y.0: udp 0 (ttl 51, id
44437) (ttl 51, id 17453)
                         4500 0044 442d 0000 3301 feaf yyyy yyyy
                         xxxx xxxx 0303 739c 0000 0000 4500 001c
                         ad95 0000 3311 955f xxxx xxxx yyyy yyyy
                         09c2 0000 0008 b13f dd2c 2a16 38e1 7646
                         7aaa 9d41
```

As it seems Foundry switches will pad 12 bytes with ICMP Port unreachable.

Other fingerptinting facts that are outlined through this section will halp us to differentiate between the operating systems, which carry the same behavior.

I have examined three ICMP Error Messages a Host can issue:

- ICMP Port Unreachable
- ICMP Protocol Unreachable
- ICMP IP Reassembly Time Exceeded

Other ICMP Error Messages, which a Host can issue and should be checked to see if they hold more fingerprinting differences, are:

- Source Quench
- Parameter Problem

6.15 LINUX ICMP Error Message Quoting Size Differences / The 20 Bytes from No Where

We must understand that there are differences between the different ICMP Error messages, not only with their meaning, but also with their implementation. I was expecting that several characters with the ICMP Error messages will be the same along all of the ICMP Error Messages, but I was wrong regarding few operating systems.

The most interesting case is with the LINUX operating system based on Kernel 2.2.x and 2.4.t-x.

The next example is with LINUX based on Kernel 2.2.16 as the targeted machine, eliciting an ICMP Port Unreachable error message:

```
00:21:30.199408 ppp0 > x.x.x.x.2066 > y.y.y.y.2000: udp 0 (ttl 64, id
1732)
                         4500 001c 06c4 0000 4011 c895 xxxx xxxx
                         yyyy yyyy 0812 07d0 0008 4484

00:21:30.493691 ppp0 < y.y.y.y > x.x.x.x: icmp: y.y.y.y udp port 2000
unreachable Offending pkt: x.x.x.x.2066 > y.y.y.y.2000: udp 0 (ttl 44,
id 1732) [tos 0xc0]   (ttl 238, id 53804)
                         45c0 0038 d22c 0000 ee01 4e60 yyyy yyyy
                         xxxx xxxx 0303 a88e 0000 0000 4500 001c
                         06c4 0000 2c11 dc95 xxxx xxxx yyyy yyyy
                         0812 07d0 0008 4484
```

The quoted data is the entire offending datagram. LINUX ICMP Error messages will be up to 576 bytes long according to the LINUX source code.

The next example is with LINUX as the targeted operating system. With this example I have sent a protocol scan with NMAP:

```
13:14:56.942897    < x.x.x.x > y.y.y.y: ip-proto-38 0 (ttl 39, id 37623)
                         4500 0014 92f7 0000 2726 02cb xxxx xxxx
                         yyyy yyyy
13:14:56.942964    > y.y.y.y > x.x.x.x: icmp: y.y.y.y protocol 38
unreachable Offending pkt: x.x.x.x > y.y.y.y: ip-proto-38 0 (ttl 39, id
37623) [tos 0xc0]   (ttl 255, id 1884)
                         45c0 0044 075c 0000 ff01 b59a yyyy yyyy
                         xxxx xxxx 0302 fb1a 0000 0000 4500 0014
                         92f7 0000 2726 02cb xxxx xxxx yyyy yyyy
                         0050 dc84 ae6f 6910 0000 0000 5004 0000
                         bd89 0000
```

LINUX adds to the entire offending packet that was quoted, another 20 bytes.

Since LINUX handles the ICMP Protocol Unreachable Error Messages like the ICMP Fragment Reassembly Time Exceeded Error Messages we will see the same pattern with ICMP Fragment Reassembly Time Exceeded:

```
[root@godfather bin]# hping2 -c 1 -x -y y.y.y.y
ppp0 default routing interface selected (according to /proc)
HPING y.y.y.y ppp0 y.y.y.y): NO FLAGS are set, 40 headers + 0 data
bytes

--- y.y.y.y hping statistic ---
1 packets tramitted, 0 packets received, 100% packet loss
round-trip min/avg/max = 0.0/0.0/0.0 ms
[root@godfather bin]#
```

The tcpdump trace:

```
19:49:22.999108 ppp0 > x.x.x.x.cvspserver > y.y.y.y.0: .
1709055398:1709055398(0) win 512 (frag 35247:20@0+) (DF) (ttl 64)
                          4500 0028 89af 6000 4006 e0ff xxxx xxxx
                          yyyy yyyy 0961 0000 65de 1da6 6a01 476b
                          5000 0200 bf71 0000

19:49:53.303196 ppp0 < y.y.y.y > x.x.x.x: icmp: ip reassembly time
exceeded Offending pkt: x.x.x.x.cvspserver > y.y.y.y.0: .
1709055398:1709055398(0) win 512 (frag 35247:20@0+) (DF) (ttl 45) [tos
0xc0]  (ttl 238, id 379)
                          45c0 0058 017b 0000 ee01 1a49 yyyy yyyy
                          xxxx xxxx 0b01 3caf 0000 0000 4500 0028
                          89af 6000 2d06 f3ff xxxx xxxx yyyy yyyy
                          0961 0000 65de 1da6 6a01 476b 5000 0200
                          bf71 0000 601d 1f0d 7a04 5045 0100 0000
                          4146 4345 4a45 4f46
```

Since LINUX's ICMP Error messages will not be bigger than 576 bytes long, if the offending packet will be big enough (not likely in real world situation) we will not see the added 20 bytes in the ICMP Fragment Reassembly / ICMP Protocol Unreachable error messages.

This unique pattern will allow us to identify LINUX based machines even if the Precedence Bits value with the LINUX ICMP Error messages will be changed to 0x000.

6.16 Foundry Networks Networking Devices Padded Bytes with ICMP Port Unreachable(s) / The 12 Bytes from No Where

Linux is not the only operating system that will have weird data bytes padded to one of his ICMP Error messages.

Foundry Network's networking devices will pad extra 12 bytes of data with their ICMP Port Unreachable Error messages. Our first example is with a ServerIron switch running software version 7.1.02T12, eliciting an ICMP Port Unreachable error message:

```
[root@godfather]# hping2 -2 -c 1 y.y.y.y
eth0 default routing interface selected (according to /proc)
HPING y.y.y.y (eth0 y.y.y.y): udp mode set, 28 headers + 0 data bytes
ICMP Port Unreachable from y.y.y.y (y.y.y.y)

--- y.y.y.y hping statistic ---
1 packets tramitted, 0 packets received, 100% packet loss
round-trip min/avg/max = 0.0/0.0/0.0 ms
[root@godfather]#


12:08:47.793503 eth0 > x.x.x.x.2498 > y.y.y.y.0: udp 0 (ttl 64, id
44437)
                          4500 001c ad95 0000 4011 885f xxxx xxxx
                          yyyy yyyy 09c2 0000 0008 b13f

12:08:48.240208 eth0 < y.y.y.y > x.x.x.x: icmp: y.y.y.y udp port 0
unreachable Offending pkt: x.x.x.x.2498 > y.y.y.y.0: udp 0 (ttl 51, id
44437) (ttl 51, id 17453)
                          4500 0044 442d 0000 3301 feaf yyyy yyyy
                          xxxx xxxx 0303 739c 0000 0000 4500 001c
```

```
                         ad95 0000 3311 955f xxxx xxxx yyyy yyyy
                         09c2 0000 0008 b13f dd2c 2a16 38e1 7646
                         7aaa 9d41
```

From the tcpdump trace we can see that the offending packet's IP header and the first 8 data bytes were echoed correctly. Right after those, 12 bytes were padded, that came from nowhere.

The next example is with Foundry Network's BigIron 8000 running software version 6.6.05T51. With this test I have sent a UDP datagram with 80 bytes of data to a closed UDP port on the BigIron 8000:

```
[root@godfather /root]# hping2 -2 -c 3 -d 80 y.y.y.y
ppp0 default routing interface selected (according to /proc)
HPING y.y.y.y  (ppp0 y.y.y.y  ): udp mode set, 28 headers + 80 data
bytes
ICMP Port Unreachable from y.y.y.y  (y.y.y.y)
ICMP Port Unreachable from y.y.y.y  (y.y.y.y)
ICMP Port Unreachable from y.y.y.y  (y.y.y.y)

--- y.y.y.y  hping statistic ---
3 packets tramitted, 0 packets received, 100% packet loss
round-trip min/avg/max = 0.0/0.0/0.0 ms
[root@godfather /root]#


11:40:36.694235 ppp0 > x.x.x.x.2779 > y.y.y.y.0: udp 80 (ttl 64, id
25211)
                         4500 006c 627b 0000 4011 2e7a xxxx xxxx
                         yyyy yyyy 0adb 0000 0058 3d09 5858 5858
                         5858 5858 5858 5858 5858 5858 5858 5858
                         5858 5858 5858 5858 5858 5858 5858 5858
                         5858 5858 5858 5858 5858 5858 5858 5858
                         5858 5858 5858 5858 5858 5858 5858 5858
                         5858 5858 5858 5858 5858 5858

11:40:37.913018 ppp0 < y.y.y.y > x.x.x.x: icmp: y.y.y.y udp port 0
unreachable Offending pkt: x.x.x.x.2779 > y.y.y.y.0: udp 80 (ttl 52, id
25211) (ttl 52, id 60504)
                         4500 0044 ec58 0000 3401 b0d4 yyyy yyyy
                         xxxx xxxx 0303 edf3 0000 0000 4500 006c
                         627b 0000 3411 3a7a xxxx xxxx yyyy yyyy
                         0adb 0000 0058 3d09 1c1d 1e1f 2021 2223
                         2425 2627
```

Again, the offending packet's IP Header and the first 8 data bytes are quoted correctly. 12 data bytes are padded right after.

A nice pattern that allows us to identify Foundry Network's networking devices.

6.17 ICMP Error Message Echoing Integrity (Tested with ICMP Port Unreachable)

When sending back an ICMP error message, some stack implementations may alter the original IP header, which is echoed back with the ICMP error message.

If an attacker examines the types of alternation that have been made to the headers, he may be able to make certain assumptions about the target operating system.

The only two field values we expect to be changed are the IP time-to-live field value and the IP header checksum. The TTL field value changes because the field is decremented by one, each time the IP Header is processed. The IP header checksum is recalculated each time the IP TTL field value decremented.

Fyodor gives the following examples in his article "Remote OS detection via TCP/IP Stack Finger Printing"[44]:

> "For example, AIX and BSDI send back an IP 'total length' field that is 20 bytes too high. Some BSDI, FreeBSD, OpenBSD, ULTRIX, and VAXen change the IP ID that you sent them. While the checksum is going to change due to the changed TTL anyway, there are some machines (AIX, FreeBSD, etc.) which send back an inconsistent or 0 checksum. Same thing goes with the UDP checksum."

This section deals with the ICMP Port Unreachable error message.

AIX 4.2.1, 4.3, 4.3 fix pack 2
In the next example I have sent a UDP datagram to a closed UDP port on an AIX 4.3 machine using HPING2. This is the tcpdump trace:

```
12:33:17.319275 ppp0 > x.x.x.x.2160 > y.y.y.y.0: udp 0 [tos 0x10]  (ttl
64, id 47349)
                         4510 001c b8f5 0000 4011 9bea xxxx xxxx
                         yyyy yyyy 0870 0000 0008 d18c

12:33:17.614823 ppp0 < y.y.y.y > x.x.x.x: icmp: y.y.y.y udp port 0
unreachable Offending pkt: x.x.x.x.2160 > y.y.y.y.0: udp 0 [tos 0x10]
(ttl 49, id 47349, bad cksum aaea!) [tos 0x10]  (ttl 241, id 17965)
                         4510 0038 462d 0000 f101 5da6 yyyy yyyy
                         xxxx xxxx 0303 f470 0000 0000 4510 0030
                         b8f5 0000 3111 aaea xxxx xxxx yyyy yyyy
                         0870 0000 0008 0000
```

We can identify several changed between the original IP Headers to the echoed ICMP Header with the ICMP port unreachable error message.

- **IP Total Length Field** - The total length field with the original UDP datagram equal to 28 bytes. With the echoed original IP header this value was changed to 48 bytes. 20 bytes more than the original UDP datagram's length.
- **IP TTL Field value** - With the ICMP error message this value is set to the value, which reached its final destination (with this example the targeted host). When it reached it target the TTL was set to 49. We also learn the target is 64-49 = 15 hops away.
- **IP Header Checksum** - The IP Header checksum was changed because the IP Total Length filed value and the IP TTL field value were changed.

[44] http://www.insecure.org/nmap/nmap-fingerprinting-article.html

- **UDP Header Checksum** – The UDP header checksum with the echoed information equal to zero.

AIX 4.1

In the next example I have sent a UDP datagram to a closed UDP port on an AIX 4.1 machine using HPING2. This is the tcpdump trace:

```
00:56:07.894612 ppp0 > x.x.x.x.1594 > y.y.y.y.0: udp 0 [tos 0x8]  (ttl
64, id 2153)
                         4508 001c 0869 0000 4011 c54f xxxx xxxx
                         yyyy yyyy 063a 0000 0008 4c93

00:56:08.204551 ppp0 < y.y.y.y > x.x.x.x: icmp: y.y.y.y udp port 0
unreachable Offending pkt: x.x.x.x.1594 > y.y.y.y.0: udp 0 [tos 0x8]
(ttl 47, id 2153, bad cksum d64f!) [tos 0x8]  (ttl 239, id 1065)
                         4508 0038 0429 0000 ef01 1a83 yyyy yyyy
                         xxxx xxxx 0303 aa13 0000 0000 4508 0030
                         0869 0000 2f11 d64f xxxx xxxx yyyy yyyy
                         063a 0000 0008 4c93
```

We can identify several changed between the original IP Headers to the echoed ICMP Header with the ICMP port unreachable error message.

- **IP Total Length Field** - The total length field with the original UDP datagram equal to 28 bytes. With the echoed original IP header this value was changed to 48 bytes. 20 bytes more than the original UDP datagram's length.
- **IP TTL Field value** - With the ICMP error message this value is set to the value, which reached its final destination (with this example the targeted host). When it reached it target the TTL was set to 47. We also learn the target is 64-47 = 17 hops away.
- **IP Header Checksum** - The IP Header checksum was changed because the IP Total Length filed value and the IP TTL field value were changed.

ICMP Error Message Echoing Integrity with different 4.x versions of AIX

In contrast to AIX version 4.3 and 4.2.1 AIX version 4.1 use the original UDP Checksum. This detail helps us to differentiate between the different versions of AIX.

BSDI 4.x

In the next example I have sent, again, a UDP datagram to a close UDP port, this time on a BSDI 4.1 machine. The following is the tcpdump trace:

```
01:01:11.128420 ppp0 > x.x.x.x.2933 > y.y.y.y.0: udp 0 [tos 0x8]  (ttl
64, id 49317)
                         4508 001c c0a5 0000 4011 9209 xxxx xxxx
                         yyyy yyyy 0b75 0000 0008 cc4e

01:01:11.484552 ppp0 < y.y.y.y.4 > x.x.x.x: icmp: y.y.y.y udp port 0
unreachable Offending pkt: x.x.x.x.2933 > y.y.y.y.0: udp 0 [tos 0x8]
(ttl 53, id 49317, bad cksum 0!) (ttl 242, id 16127)
                         4500 0038 3eff 0000 f201 61ab yyyy yyyy
```

```
                    xxxx xxxx 0303 c226 0000 0000 4508 0030
                    c0a5 0000 3511 0000 xxxx xxxx yyyy yyyy
                    0b75 0000 0008 cc4e
```

Again several changed were made to the original IP Header: .

- **IP Total length** - With the echoed IP Header this field value was changed from the original 28 bytes to 48 bytes. 20 bytes more than the original.
- **IP TTL Field Value** – Changed according to the hop count. Was equal to 53 when arrived to its destination. The target is 64 – 53 = 11 hops away.
- **IP Header Checksum** – changed now it equal to zero!

FreeBSD 3.x up to 4.1.1 (not including)
The next example is with FreeBSD 4.1:

```
00:52:19.055758 ppp0 > x.x.x.x.1393 > y.y.y.y.0: udp 0 [tos 0x8]  (ttl
64, id 58965)
                    4508 001c e655 0000 4011 3f63 xxxx xxxx
                    yyyy yyyy 0571 0000 0008 a55c

00:52:19.464548 ppp0 < y.y.y.y > x.x.x.x: icmp: y.y.y.y udp port 0
unreachable Offending pkt: x.x.x.x.1393 > y.y.y.y.0: udp 0 [tos 0x8]
(ttl 47, id 21990, bad cksum 5063!) (ttl 238, id 27639)
                    4500 0038 6bf7 0000 ee01 0bbd yyyy yyyy
                    xxxx xxxx 0303 87f3 0000 0000 4508 001c
                    55e6 0000 2f11 5063 xxxx xxxx yyyy yyyy
                    0571 0000 0008 0000
```

- The **IP Identification** field value is changed. This field is constructed with 16bit. The first 8 bits changed places with the second pair of 8 bits constructing this field. With the original datagram this field value was e655, with the echoed IP header it is 55e6[45].
- The **IP TTL field value** has changed. The target is 64 – 47 = 17 hops away.
- The **IP Header Checksum** have changed because of the parameters were changed as well.
- The **UDP checksum** is changed and now it equal to zero!

Operating System	DF Bit set with the Reply?	IP Total Length	IP Identification	IP TTL field value	IP Header Checksum	UDP Checksum
LINUX Kernel 2.2.x	No	Same	Same	Changed according to hop count.	Changed because of new parameters.	Same
LINUX Kernel 2.4	No	Same	Same	Changed according to hop count.	Changed because of new parameters.	Same

[45] http://www.freebsd.org/cgi/query-pr.cgi?pr=16240 ; Patches were issued. This is fixed with FreeBSD 4.1.1.

FreeBSD 4.0	No	Same	Changed. The first two bits are flipped with the second pair. Gives a new value.	Changed according to hop count.	Changed because of new parameters.	Changed. Now equal to ZERO!
FreeBSD 4.11	No	Same	Same	Changed according to hop count.	Changed because of new parameters.	Changed. Now equal to ZERO!
BSDI 4.1	No	Changed (20 bytes more)	Same	Changed according to hop count	Changed. Now equals to ZERO!	Same
Sun Solaris 2.6	Yes	Same	Same	Changed according to hop count.	Changed because of new parameters.	Same
Sun Solaris 2.7	Yes	Same	Same	Changed according to hop count.	Changed because of new parameters.	Same
Sun Solaris 2.8[46]	Yes	Same	Same	Changed according to hop count.	Changed because of new parameters.	Same
HPUX 11.0	No -> Yes	Same	Same	Changed according to hop count.	Changed because of new parameters.	Same
Compaq Tru64	No	Same	Same	Changed according to hop count.	Changed because of new parameters.	Changed. Now equal to ZERO!
DG-UX 5.6	No	Same	Same	Changed according to hop count.	Changed because of new parameters.	Changed. Now equal to ZERO!
AIX 4.3 fp2, 4.3, 4.2.1	No	Changed (20 bytes more)	Same	Changed according to hop count	Changed because of new parameters.	Changed. Now equal to ZERO!
AIX 4.1	No	Changed (20 bytes more)	Same	Changed according to hop count	Changed because of new parameters.	Same

[46] The DF Bit is set.

Operating System	DF Bit set with the Reply?	IP Total Length	IP Identificatio n	IP TTL field value	IP Header Checksum	UDP Checksum
ULTRIX	No	Same	Changed. The first two bits are flipped with the second pair. Gives a new value.	Changed according to hop count	Changed. Now equals to ZERO!	Changed. Now equal to ZERO!
OpenVMS	No	Same	Changed. The first two bits are flipped with the second pair. Gives a new value.	Changed according to hop count	Changed. Now equals to ZERO!	Changed. Now equal to ZERO!
Microsoft windows 98						
Mirosoft Windows 98SE	No	Same	Same	Changed according to hop count.	Changed because of new parameters.	Same
Microsoft Windows ME	No	Same	Same	Changed according to hop count.	Changed because of new parameters.	Same
Microsoft Windows NT 4	No	Same	Same	Changed according to hop count.	Changed because of new parameters.	Same
Microsoft Windows 2000 Family	No	Same	Same	Changed according to hop count.	Changed because of new parameters.	Same

Table 14: ICMP Error Message Echoing Integrity

6.18 Novell Netware Echoing Integrity Bug with ICMP Fragment Reassembly Time Exceeded

Novell Netware operating systems have a unique pattern with ICMP Fragment Reassembly Time Exceeded error messages they produce.

In general, when an ICMP error message is produced, the offending packet's IP Header + at least 8 bytes of data are echoed with the error message.

If we examine closely the next example, we can see that the Offending packet's IP TTL field value echoed back is zero.

We expect this value to decrement from the value initially assigned; but not to be zero. Since this value should change from one hop to another, the Checksum need to be recalculated each time. With the Novell Netware error message we can see that the Checksum echoed is miscalculated.

...And again this is a Fragment Reassembly Time Exceeded ICMP error message and not an ICMP Time Exceeded in Transit error message.

The next example is with Novell Netware 5.1:

```
[root@godfather bin]# hping2 -c 1 -x -y y.y.y.y
ppp0 default routing interface selected (according to /proc)
HPING y.y.y.y (ppp0 y.y.y.y): NO FLAGS are set, 40 headers + 0 data
bytes

--- y.y.y.y hping statistic ---
1 packets tramitted, 0 packets received, 100% packet loss
round-trip min/avg/max = 0.0/0.0/0.0 ms
[root@godfather bin]#
```

The Trace:

```
20:12:28.008893 ppp0 > x.x.x.x.1865 > y.y.y.y.0: .
687160929:687160929(0) win 512 (frag 58586:20@0+) (DF) (ttl 64)
                         4500 0028 e4da 6000 4006 c236 xxxx xxxx
                         yyyy yyyy 0749 0000 28f5 3e61 669e 9f15
                         5000 0200 c5d2 0000

20:12:41.313202 ppp0 < y.y.y.y > x.x.x.x: icmp: ip reassembly time
exceeded Offending pkt: [|tcp] (frag 58586:20@0+) (DF) [ttl 0] (bad
cksum d3361) (ttl 111, id 9591)
                         4500 0038 2577 0000 6f01 b28f yyyy yyyy
                         xxxx xxxx 0b01 b55f 0000 0000 4500 0028
                         e4da 6000 0006 d336 xxxx xxxx yyyy yyyy
                         0749 0000 28f5 3e61
```

This unique pattern enables us to determine if the operating system in question is a Novell Netware or other with one datagram only.

6.19 The Precedence bits with ICMP Error Messages (Identifying LINUX)

Each IP Datagram has an 8-bit field called the "TOS Byte", which represents the IP support for prioritization and Type-of-Service handling.

<table>
<tr><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td></tr>
<tr><td colspan="3">Precedence</td><td colspan="4">TOS</td><td>MBZ</td></tr>
</table>

Figure 13: The Type of Service Byte

The "TOS Byte" consists of three fields.

The "Precedence field", which is 3-bit long, is intended to prioritize the IP Datagram. It has eight levels of prioritization[47]:

Precedence	Definition
0	Routine (Normal)
1	Priority
2	Immediate
3	Flash
4	Flash Override
5	Critical
6	Internetwork Control
7	Network control

Table 15: Precedence Field Values

Higher priority traffic should be sent before lower priority traffic.

The second field, 4 bits long, is the "Type-of-Service" field. It is intended to describe how the network should make tradeoffs between throughput, delay, reliability, and cost in routing an IP Datagram.

The last field, the "MBZ" (most be zero), is unused and most be zero. Routers and hosts ignore this last field. This field is 1 bit long.

RFC 1122 Requirements for Internet Hosts – Communication Layers, states:
"The Precedence field is intended for Department of Defense applications of the Internet protocols. The use of non-zero values in this field is outside the scope of this document and the IP standard specification. Vendors should consult the Defense Communication Agency (DCA) for guidance on the IP Precedence field and its implications for other protocol layers. However, vendors should note that the use of precedence will most likely require that its value be passed between protocol layers in just the same way as the TOS field is passed".

Other precedence information is available with RFC 1812 Requirements for IP Version 4 Routers:
"4.3.2.5 TOS and Precedence
...

ICMP Source Quench error messages, if sent at all, MUST have their IP Precedence field set to **the same value as the IP Precedence field in the packet that provoked the sending of the ICMP Source Quench message.** All other ICMP error messages (Destination Unreachable, Redirect, Time Exceeded, and Parameter Problem) SHOULD have their precedence value set to 6 (INTERNETWORK CONTROL) or 7 (NETWORK CONTROL). The IP Precedence value for these error messages MAY be settable".

With the operating systems I have checked, nearly all of them used the value of 0x00 for the Precedence field (bits).

All but LINUX

[47] RFC 791 – Internet Protocol, http://www.ietf.org/rfc/rfc791.txt

Fyodor had outlined in his paper "Remote OS Identification by TCP/IP Fingerprinting"[48] the fact that LINUX is using the value of 0xc0 (an unused precedence value) as its TOS byte value with ICMP Port Unreachable error messages.

In the next example we have sent one UDP packet destined to port 50 (which is closed on the destination machine) from one LINUX machine to another, both running Redhat LINUX 6.1:

```
[root@stan /root]# hping2 -2 192.168.5.5 -p 50 -c 1
default routing not present
HPING 192.168.5.5 (eth0 192.168.5.5): udp mode set, 28 headers + 0 data
bytes
ICMP Port Unreachable from 192.168.5.5  (kenny.sys-security.com)

--- 192.168.5.5 hping statistic ---
1 packets tramitted, 0 packets received, 100% packet loss
round-trip min/avg/max = 0.0/0.0/0.0 ms
```

```
Kernel filter, protocol ALL, raw packet socket
Decoding Ethernet on interface eth0
03/12-12:54:47.274096 192.168.5.1:2420 -> 192.168.5.5:50
UDP TTL:64 TOS:0x0 ID:57254
Len: 8

03/12-12:54:47.274360 192.168.5.5 -> 192.168.5.1
ICMP TTL:255 TOS:0xC0 ID:0
DESTINATION UNREACHABLE: PORT UNREACHABLE
00 00 00 00 45 00 00 1C DF A6 00 00 40 11 0F D4   ....E.......@...
C0 A8 05 01 C0 A8 05 05 09 74 00 32 00 08 6A E1   .........t.2..j.
```

This abnormality with LINUX is not only limited to ICMP Destination Unreachable Port Unreachable error messages.

Lets examine the next trace:

```
00:30:08.339498 < x.x.x.x > y.y.y.y: ip-proto-72 0 (ttl 49, id 38624)
                     4500 0014 96e0 0000 3148 f4bf xxxx xxxx
                     yyyy yyyy

00:30:08.339559 > y.y.y.y > x.x.x.x: icmp: y.y.y.y protocol 72
unreachable Offending pkt: x.x.x.x > y.y.y.y: ip-proto-72 0 (ttl 49, id
38624) [tos 0xc0]    (ttl 255, id 37)
                     45c0 0044 0025 0000 ff01 bcd1 yyyy yyyy
                     xxxx xxxx 0302 fb1a 0000 0000 4500 0014
                     96e0 0000 3148 f4bf xxxx xxxx yyyy yyyy
                     0050 d909 621b 96f7 0000 0000 5004 0000
                     df71 0000
```

The ICMP error message produced by a LINUX machine based on Kernel 2.2.14, is Destination Unreachable Protocol Unreachable (Type 3 Code 2). As it can be seen the TOS Byte value that was used is again 0xc0. Which is an unused Precedence bits value.

[48] This fact was discovered by Fyodor. http://www.insecure.org/nmap/nmap-fingerprinting-article.html

LINUX embraced the behavior RFC 1812 suggested and sends all his ICMP error messages with the Precedence field value sent to 0xc0 (value of 6).

Just to remind the reader – LINUX is not a router.

6.20 TOS Bits (=field) Echoing with ICMP Error

RFC 1394 specify that an ICMP error message be always sent with the default TOS field value of 0000 (TOS field=TOS bits in the TOS Byte).

When an offending packet with a TOS field value of 0000 is eliciting an ICMP error message from an offended host, the TOS field value with all the operating systems I have checked will be set to 0000.

If we will pay attention to the TOS Byte we will see that LINUX and several routers will use the value of 0xc0 for the precedence field (see section 6.14 The Precedence bits with ICMP Error Messages for the explanation).

What will happen if the TOS field with the offending packet will be set to a value different than the default (0000)?

We will have several operating systems that will echo the TOS field back with the ICMP error message.

Our first example is with an AIX 4.3 machine, where a UDP datagram is sent with a TOS field value of 0x10 hex:

```
12:33:17.319275 ppp0 > x.x.x.x.2160 > y.y.y.y.0: udp 0 [tos 0x10]  (ttl
64, id 47349)
                         4510 001c b8f5 0000 4011 9bea xxxx xxxx
                         yyyy yyyy 0870 0000 0008 d18c
```

```
12:33:17.614823 ppp0 < y.y.y.y > x.x.x.x: icmp: y.y.y.y udp port 0
unreachable Offending pkt: x.x.x.x.2160 > y.y.y.y.0: udp 0 [tos 0x10]
(ttl 49, id 47349, bad cksum aaea!) [tos 0x10]  (ttl 241, id 17965)
                         4510 0038 462d 0000 f101 5da6 yyyy yyyy
                         xxxx xxxx 0303 f470 0000 0000 4510 0030
                         b8f5 0000 3111 aaea xxxx xxxx yyyy yyyy
                         0870 0000 0008 0000
```

As it can be seen from the trace, the TOS field value was echoed back by the AIX machine. This was tested against AIX 4.1, 4.2.1, 4.3, 4.3 fix pack2.

The next example is with DGUX 5.6:

```
12:58:57.663517 ppp0 > x.x.x.x.1074 > y.y.y.y.11: udp 0 [tos 0x8]  (ttl
64, id 47314)
                         4508 001c b8d2 0000 4011 a037 xxxx xxxx
                         yyyy yyyy 0432 000b 0008 d9e1
```

```
12:58:57.984820 ppp0 < 134.210.1.200 > x.x.x.x.: icmp: y.y.y.y.200 udp
port 11 unreachable Offending pkt: x.x.x.x.1074 > y.y.y.y.11: udp 0
[tos 0x8]  (ttl 52, id 47314) [tos 0x8]  (ttl 52, id 16984)
```

```
4508 0038 4258 0000 3401 22a6 yyyy yyyy
d508 c41c 0303 f8b7 0000 0000 4508 001c
b8d2 0000 3411 ac37 xxxx xxxx yyyy yyyy
0432 000b 0008 0000
```

How can we differentiate between DGUX and AIX? If we will pay attention to the echoing integrity. AIX 4.x sets the IP total length field value, with the echoed offending IP Header, to a value 20 bytes higher than the original. DGUX quote this field value correctly.

The last operating system, which I have found echoing the TOS field value with its ICMP error messages, is LINUX operating systems based on Kernel 2.2.x & 2.4 (the versions of the Kernel that I have tested):

```
00:50:43.759906 ppp0 > x.x.x.x.1952 > y.y.y.y.0: udp 0 [tos 0x10]  (ttl
64, id 15952)
                        4510 001c 3e50 0000 4011 e6b2 xxxx xxxx
                        yyyy yyyy 07a0 0000 0008 a27f

00:50:44.154556 ppp0 < y.y.y.y > x.x.x.x: icmp: y.y.y.y.211 udp port 0
unreachable Offending pkt: x.x.x.x.1952 > y.y.y.y.0: udp 0 [tos 0x10]
(ttl 47, id 15952) [tos 0xd0]  (ttl 238, id 54662)
                        45d0 0038 d586 0000 ee01 a0af yyyy yyyy
                        xxxx xxxx 0303 52d5 0000 0000 4510 001c
                        3e50 0000 2f11 f7b2 xxxx xxxx yyyy yyyy
                        07a0 0000 0008 a27f
```

Another unique pattern with LINUX is setting the Precedence field value to 0xc0 with ICMP error messages. This helps us to differentiate LINUX from the other operating systems that echo the TOS field value.

While LINUX embraced RFC 1812 instructions for routers regarding the TOS and Precedence fields, the other operating systems that echo the TOS field value don't seem to have a good excuse for doing so.

6.21 DF Bit Echoing with ICMP Error Messages

We already have the DF Bit Echoing method with ICMP Query message types (& Replies); I was thinking why this couldn't happen with ICMP Error Messages as well?

What will happen if we will set the DF bit with an offending packet that will generate an ICMP Error message? Will the DF Bit be set with the ICMP Error Message?

In the next example, a UDP datagram is sent to a closed UDP port, to elicit an ICMP Port Unreachable error message. The DF bit is set with the offending datagram. As it can be seen the DF bit is set with the ICMP error message the FreeBSD 4.1.1 machine, which was the target system issued back.

```
[root@godfather /root]# hping2 -2 -p 2000 -c 2 -y y.y.y.y
ppp0 default routing interface selected (according to /proc)
HPING y.y.y.y (ppp0 y.y.y.y): udp mode set, 28 headers + 0 data bytes
ICMP Port Unreachable from y.y.y.y  (host_address)
ICMP Port Unreachable from y.y.y.y  (host_address)

--- y.y.y.y hping statistic ---
```

```
2 packets tramitted, 0 packets received, 100% packet loss
round-trip min/avg/max = 0.0/0.0/0.0 ms
[root@godfather /root]#

00:31:29.805075 ppp0 > x.x.x.x.1403 > y.y.y.y.2000: udp 0 (DF) (ttl 64,
id 19417)
                    4500 001c 4bd9 4000 4011 452b xxxx xxxx
                    yyyy yyyy 057b 07d0 0008 48c6

00:31:30.103692 ppp0 < 18.170.1.79 > x.x.x.x: icmp: y.y.y.y udp port
2000 unreachable Offending pkt: x.x.x.x.1403 > y.y.y.y.2000: udp 0 (DF)
(ttl 45, id 19417) (DF) (ttl 238, id 47017)
                    4500 0038 b7a9 4000 ee01 2b4e yyyy yyyy
                    xxxx xxxx 0303 efa9 0000 0000 4500 001c
                    4bd9 4000 2d11 582b xxxx xxxx yyyy yyyy
                    057b 07d0 0008 0000
```

We can distinguish between the group of operating systems, which will echo back the DF bit with their replies, to the group of operating systems that will not.

The next example is with Microsoft Windows ME:

```
00:49:45.853751 ppp0 > x.x.x.x.1580 > y.y.y.y.10: udp 0 (DF) (ttl 64,
id 63227)
                    4500 001c f6fb 4000 4011 730a xxxx xxxx
                    yyyy yyyy 062c 000a 0008 28dd

00:49:46.173681 ppp0 < 212.150.102.96 > x.x.x.x: icmp: y.y.y.y udp port
10 unreachable Offending pkt: x.x.x.x.1580 > y.y.y.y.10: udp 0 (DF)
(ttl 55, id 63227) (ttl 119, id 430)
                    4500 0038 01ae 0000 7701 714c yyyy yyyy
                    xxxx xxxx 0303 cde1 0000 0000 4500 001c
                    f6fb 4000 3711 7c0a xxxx xxxx yyyy yyyy
                    062c 000a 0008 28dd
```

Among the operating systems I have checked LINUX machines based on Kernel 2.2.x / 2.4.x, ULTRIX, Novell Netware, and Microsoft Windows 98/98SE/ME/NT4SP6A/Windows 2000 Family, will not echo back the DF bit with their ICMP Error messages.

How can we distinguish between the operating systems in the non-DF echoing group?
Since Linux is using the value of 0xc0 hex for his Precedence Bits field value for all ICMP Error messages we can separate it instantly.

```
00:25:17.203727 ppp0 > x.x.x.x.1421 > y.y.y.y.2000: udp 0 (DF) (ttl 64,
id 11969)
                    4500 001c 2ec1 4000 4011 b938 xxxx xxxx
                    yyyy yyyy 058d 07d0 0008 9fa9

00:25:17.573698 ppp0 < y.y.y.y > x.x.x.x: icmp: y.y.y.y udp port 2000
unreachable Offending pkt: x.x.x.x.1421 > y.y.y.y.2000: udp 0 (DF) (ttl
45, id 11969) [tos 0xc0]  (ttl 236, id 38250)
                    45c0 0038 956a 0000 ec01 e5c2 yyyy yyyy
                    xxxx xxxx 0303 4fee 0000 0000 4500 001c
                    2ec1 4000 2d11 cc38 xxxx xxxx yyyy yyyy
                    058d 07d0 0008 9fa9
```

ULTRIX echo integrity is not that good. The offending packet echoing will set both the IP Header Checksum and the Original UDP Checksum to zero. It will also miscalculate the IP ID field value and will flip the first 8 bits with the second one, creating a false value for it:

```
00:29:05.013726 ppp0 > x.x.x.x.1188 > y.y.y.y.2000: udp 0 (DF) (ttl 64,
id 34921)
                     4500 001c 8869 4000 4011 5f85 xxxx xxxx
                     yyyy yyyy 04a4 07d0 0008 a087

00:29:05.383686 ppp0 < 194.47.250.222 > x.x.x.x: icmp: y.y.y.y udp port
2000 unreachable Offending pkt: x.x.x.x.1188 > y.y.y.y.2000: udp 0 (ttl
45, id 27016, bad cksum 0!) (ttl 236, id 9736)
                     4500 0038 2608 0000 ec01 55da yyyy yyyy
                     xxxx xxxx 0303 c1e7 0000 0000 4500 001c
                     6988 0000 2d11 0000 xxxx xxxx yyyy yyyy
                     04a4 07d0 0008 0000
```

This will leave us with Novell Netware and the various Microsoft Windows Operating Systems.

As discussed in section 6.17 "*Novell Netware Echoing Integrity Bug with ICMP Fragment Reassembly Time Exceeded*", when a Novell Netware operating system issue an ICMP Time Exceeded error message it will zero out on the echoed offending packet the IP TTL field value. We will use this information and send an offending packet to the questioned operating systems that will elicit an ICMP Time Exceeded error message from the quesitoned OSs.

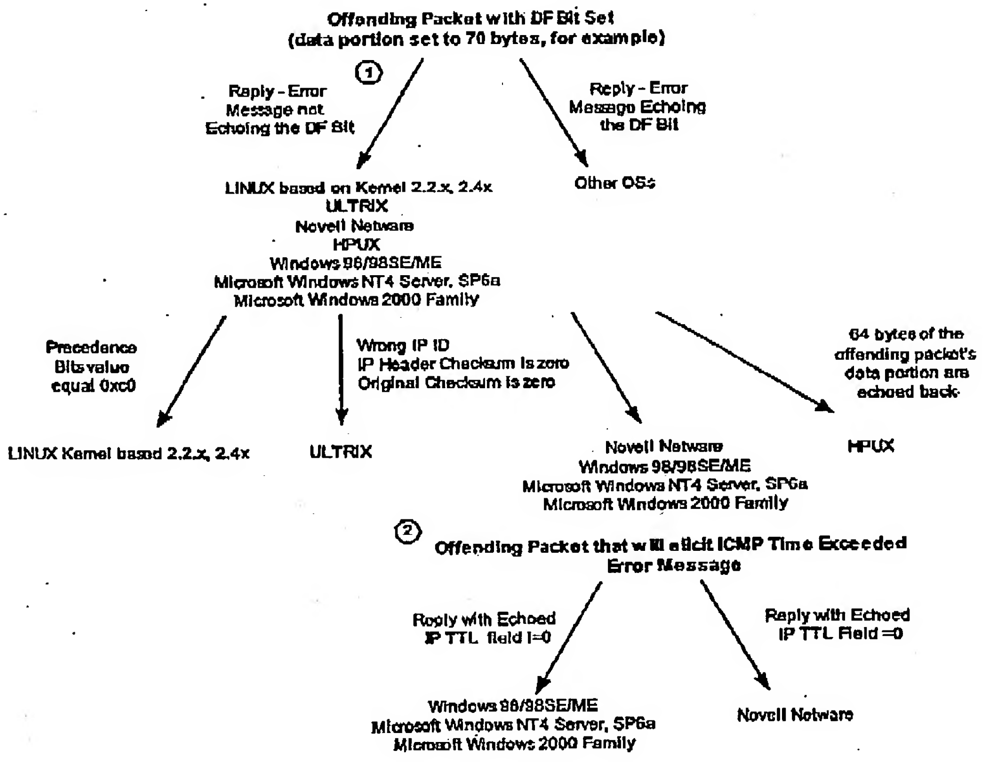

Offending Packet with DF Bit Set
(data portion set to 70 bytes, for example)
1
Reply - Error Message not Echoing the DF Bit
Reply - Error Message Echoing the DF Bit
LINUX based on Kernel 2.2.x, 2.4x
ULTRIX
Novell Netware
HPUX
Windows 98/98SE/ME
Microsoft Windows NT4 Server, SP6a
Microsoft Windows 2000 Family
Other OSs
Precedence Bits value equal 0xc0
Wrong IP ID
IP Header Checksum is zero
Original Checksum is zero
64 bytes of the offending packet's data portion are echoed back
LINUX Kernel based 2.2.x, 2.4x
ULTRIX
Novell Netware
Windows 98/98SE/ME
Microsoft Windows NT4 Server, SP6a
Microsoft Windows 2000 Family
HPUX
2
Offending Packet that will elicit ICMP Time Exceeded Error Message
Reply with Echoed IP TTL field !=0
Reply with Echoed IP TTL Field =0
Windows 98/98SE/ME
Microsoft Windows NT4 Server, SP6a
Microsoft Windows 2000 Family
Novell Netware

Diagram 12: DF Bit Echoing with ICMP Error Messages

We can take a second approach using the ICMP Fragment Reassembly Time Exceeded error message. We will send an offending packet with the DF bit set that will elicit an ICMP Fragment Reassembly Time Exceeded error message back from the targeted machine. Novell Netware, LINUX based Kernel 2.2.x and 2.4x-t, and the various Microsoft Windows operating systems will set the DF bit with their replies. LINUX and Novell have their unique fingerprinting with ICMP Fragment Reassembly Time Exceeded error messages, enabling us to isolate the Microsoft based operating systems machines.

HP-UX 11.x based machines will have a unique behavior when the PMTU discovery process based on ICMP Echo Requests is enabled (by default). In the next example I have sent a UDP datagram to port 53 (DNS) of the targeted HPUX machine.

```
[root@godfather /root]# hping2 -2 -p 53 -c 2 -y y.y.y.y
ppp0 default routing interface selected (according to /proc)
HPING y.y.y.y (ppp0 y.y.y.y): udp mode set, 28 headers + 0 data bytes
ICMP Port Unreachable from y.y.y.y (unknown host name)
ICMP Port Unreachable from y.y.y.y (unknown host name)

--- y.y.y.y hping statistic ---
2 packets tramitted, 0 packets received, 100% packet loss
round-trip min/avg/max = 0.0/0.0/0.0 ms
[root@godfather /root]#
```

```
00:45:02.490445 ppp0 > x.x.x.x.codasrv > y.y.y.y.domain: 0 [0q] (0)
(DF) (ttl 64, id 7454)
                     4500 001c 1d1e 4000 4011 e708 xxxx xxxx
                     yyyy yyyy 0980 0035 0008 bf7e
```

As an instant reply the PMTU discovery process which is based upon ICMP Echo request(s) is started:

```
00:45:03.113686 ppp0 < y.y.y.y > x.x.x.x: icmp: echo request (DF) (ttl
242, id 25153)
                     4500 05dc 6241 4000 f201 ea34 yyyy yyyy
                     xxxx xxxx 0800 7e52 9abc def0 0000 0000
                     0000 0000 0000 0000 0000 0000 0000 0000
                     0000 0000 0000 0000 0000 0000 0000 0000
                     0000 0000 0000 0000 0000 0000 0000 0000
                     0000 0000 0000 0000 0000 0000 0000 0000
                     0000 0000 0000 0000 0000 0000 0000 0000
                     0000 0000 0000 0000 0000 0000 0000 0000
                     0000 0000 0000 0000 0000 0000 0000 0000
                     0000 0000 0000 0000 0000 0000 0000 0000
                     0000 0000 0000 0000 0000 0000 0000 0000
                     0000 0000 0000 0000 0000 0000 0000 0000
                     0000 0000 0000 0000 0000 0000 0000 0000
                     0000 0000 0000 0000 0000 0000 0000 0000
                     0000 0000 0000 0000 0000 0000 0000 0000
                     0000 0000 0000 0000 0000 0000 0000 0000
                     0000 0000 0000 0000 0000 0000 0000 0000
                     0000 0000 0000 0000 0000 0000 0000 0000
                     0000 0000 0000 0000 0000 0000 0000 0000
                     0000 0000 0000 0000 0000 0000 0000 0000
```

```
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
```

```
         0000 0000 0000 0000 0000 0000 0000 0000
         0000 0000 0000 0000 0000 0000 0000 0000
         0000 0000 0000 0000 0000 0000 0000 0000
         0000 0000 0000 0000 0000 0000 0000 0000
         0000 0000 0000 0000 0000 0000 0000 0000
         0000 0000 0000 0000 0000 0000 0000 0000
         0000 0000 0000 0000 0000 0000 0000 0000
         0000 0000 0000 0000 0000 0000 0000 0000
         0000 0000 0000 0000 0000 0000 0000 0000
         0000 0000 0000 0000 0000 0000 0000 0000
         0000 0000 0000 0000 0000 0000 0000 0000
         0000 0000 0000 0000 0000 0000 0000 0000
         0000 0000 0000 0000 0000 0000 0000 0000
         0000 0000 0000 0000 0000 0000 0000 0000
         0000 0000 0000 0000 0000 0000 0000 0000
         0000 0000 0000 0000 0000 0000 0000 0000
         0000 0000 0000 0000 0000 0000
```

My LINUX machine replied with ICMP Echo Reply:

```
00:45:03.113787 ppp0 > x.x.x.x > y.y.y.y: icmp: echo reply (ttl 255, id
98)
         4500 05dc 0062 0000 ff01 7f14 xxxx xxxx
         yyyy yyyy 0000 8652 9abc def0 0000 0000
         0000 0000 0000 0000 0000 0000 0000 0000
         0000 0000 0000 0000 0000 0000 0000 0000
         0000 0000 0000 0000 0000 0000 0000 0000
         0000 0000 0000 0000 0000 0000 0000 0000
         0000 0000 0000 0000 0000 0000 0000 0000
         0000 0000 0000 0000 0000 0000 0000 0000
         0000 0000 0000 0000 0000 0000 0000 0000
         0000 0000 0000 0000 0000 0000 0000 0000
         0000 0000 0000 0000 0000 0000 0000 0000
         0000 0000 0000 0000 0000 0000 0000 0000
         0000 0000 0000 0000 0000 0000 0000 0000
         0000 0000 0000 0000 0000 0000 0000 0000
         0000 0000 0000 0000 0000 0000 0000 0000
         0000 0000 0000 0000 0000 0000 0000 0000
         0000 0000 0000 0000 0000 0000 0000 0000
         0000 0000 0000 0000 0000 0000 0000 0000
         0000 0000 0000 0000 0000 0000 0000 0000
         0000 0000 0000 0000 0000 0000 0000 0000
         0000 0000 0000 0000 0000 0000 0000 0000
         0000 0000 0000 0000 0000 0000 0000 0000
         0000 0000 0000 0000 0000 0000 0000 0000
         0000 0000 0000 0000 0000 0000 0000 0000
         0000 0000 0000 0000 0000 0000 0000 0000
         0000 0000 0000 0000 0000 0000 0000 0000
         0000 0000 0000 0000 0000 0000 0000 0000
         0000 0000 0000 0000 0000 0000 0000 0000
         0000 0000 0000 0000 0000 0000 0000 0000
         0000 0000 0000 0000 0000 0000 0000 0000
         0000 0000 0000 0000 0000 0000 0000 0000
         0000 0000 0000 0000 0000 0000 0000 0000
         0000 0000 0000 0000 0000 0000 0000 0000
         0000 0000 0000 0000 0000 0000 0000 0000
         0000 0000 0000 0000 0000 0000 0000 0000
```

```
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
```

```
                    0000 0000 0000 0000 0000 0000 0000 0000
                    0000 0000 0000 0000 0000 0000
```

The first ICMP Port Unreachable error message arrives without the DF bit set:

```
00:45:03.123692 ppp0 < y.y.y.y > x.x.x.x: icmp: y.y.y.y udp port domain
unreachable Offending pkt: x.x.x.x.codasrv > y.y.y.y.domain: 0 [0q] (0)
(DF) (ttl 51, id 7454) (ttl 242, id 25154)
                    4500 0038 6242 0000 f201 2fd8 yyyy yyyy
                    xxxx xxxx 0303 33c1 0000 0000 4500 001c
                    1d1e 4000 3311 f408 xxxx xxxx yyyy yyyy
                    0980 0035 0008 bf7e
```

A second UDP datagram is sent:

```
00:45:03.493752 ppp0 > x.x.x.x.codasrv-se > y.y.y.y.domain: 56810+ (0)
(DF) (ttl 64, id 59904)
                    4500 001c ea00 4000 4011 1a26 xxxx xxxx
                    yyyy yyyy 0981 0035 0008 bf7d
```

The ICMP Port Unreachable error message that was sent for the second UDP datagram now sets the DF bit as part of the PMTU discovery process maintenance:

```
00:45:03.813687 ppp0 < y.y.y.y > x.x.x.x: icmp: y.y.y.y udp port domain
unreachable Offending pkt: x.x.x.x.codasrv-se > y.y.y.y.domain: 26990
op5+ [b2&3=0x2d61] [29188a] [25700q] [24946n] [28769au] (0) (DF) (ttl
51, id 59904) (DF) (ttl 242, id 25155)
                    4500 0038 6243 4000 f201 efd6 yyyy yyyy
                    xxxx xxxx 0303 33c1 0000 0000 4500 001c
                    ea00 4000 3311 2726 xxxx xxxx yyyy yyyy
                    0981 0035 0008 bf7d
```

This also means that with the regular behavior with HPUX 11.x it will not echo back the DF bit. Also, if you are sending only one offending datagram to the targeted HPUX 11.x based machine, you will not be able to see the change.

So how can we distinguish HPUX from the other operating systems?

HPUX based operating system(s) machines will echo up to 64 bytes of the offending packet's data portion. By sending a bigger offending datagram (for example with 80 bytes of data portion) we can examine which of the operating systems in question, which do not set the DF bit with the ICMP error message, will echo 64 bytes of the data portion (or even more than 8 and will not set the the precedence bits to 0xc0).

Not that useful fingerprinting method(s)

6.22 Unusual Big ICMP Echo Request

What would happen if we would send unusual big ICMP Echo message that would require its fragmentation? Would the queried operating systems will process the query correctly and produce an accurate reply?

```
[root@aik /root]# ping -s 1500 x.x.x.x
```

```
PING x.x.x.x (x.x.x.x) from y.y.y.y : 1500(1528) bytes of data.
1508 bytes from x.x.x.x: icmp_seq=0 ttl=241 time=1034.7 ms
1508 bytes from host_address (x.x.x.x): icmp_seq=2 ttl=241 time=1020.0
ms
1508 bytes from host_address (x.x.x.x): icmp_seq=3 ttl=241 time=1090.4
ms
1508 bytes from host_address (x.x.x.x): icmp_seq=5 ttl=241 time=1060.0
ms

--- x.x.x.x ping statistics ---
8 packets transmitted, 5 packets received, 37% packet loss
round-trip min/avg/max = 1000.2/1041.0/1090.4 ms
[root@aik /root]#
```

As it seems all the probed operating systems I have tested behaved correctly processing the query and sending the reply back.

What else can assist us with this kind of query?
The DF (Don't Fragment) bit.

Some operating systems would process the query and set the don't fragment bit on the fragments of the reply like we have outlined in the "DF Bit Playground" section. These operating systems would be Sun Solaris, and HP-UX 10.30 & 11.0x[49].

We can use other methods, which does not generate the kind of noise this method generates. Basically there is no reason for this size of ICMP Echo request. This should trigger IDS systems immediately that something suspicious is happening.

[49] Please refer to section 6.2 for more information.

7.0 Filtering ICMP on your Filtering Device to Prevent Scanning Using ICMP

7.1 Inbound

An example of incoming ICMP traffic that should be blocked in order to *prevent scanning techniques that were outlined in this paper* might be:

- ICMP Echo (used for Host Detection, traceroute, Inverse Mapping, and Operating System Fingerprinting).
- ICMP Echo Reply (used for Inverse Mapping)
- ICMP Time Stamp Requests (used for Host Detection, Operating System Fingerprinting)
- ICMP Address Mask Request (used for Host Detection, Operating System Fingerprinting)
- All ICMP Message Types (Inverse Mapping Technique)
- ICMP Error messages (Operating System Fingerprinting)
- All ICMP Message Types should be blocked in order to prevent the fingerprinting techniques I have outlined in this research paper.

- You should also block the IP directed broadcast on your border router.
- Deny access to your Broadcast and Network addresses from the Internet.

If you look closely at this list, it is all ICMP Message types, whether query types or error types.

7.2 Outbound

There are people who claim that any traffic type of ICMP should be allowed from a protected network to the Internet. This is not true. Filtering the incoming traffic does not mean we are protected from some of the security hazards I outlined in this paper.

7.2.1 ICMP ECHO Reply (Type 0)
Used to map a host using Host Detection.

7.2.2 ICMP Destination Unreachable Messages
I have demonstrated that host detection can be done with bad IP Header packets, which elicit various ICMP Parameter Problem and ICMP Destination Unreachable error messages from the probed machines and draw the attacked network topology.

7.2.3 ICMP "Fragmentation Needed and Don't Fragment Bit was Set"
See section 3.5

7.2.4 ICMP ECHO (Type 8)
We have to have a Stateful filtering device that would perform Stateful inspection with ICMP in order to let ICMP ECHO Requests out, and receive only the corresponding ICMP ECHO Replies.

The current state with filtering devices is not that bright. Most of them do not perform Stateful inspection with the ICMP protocol. Allowing ICMP ECHO Replies inside our protected network is very dangerous and is not worth it.

Unless you use a Stateful filtering device with the ICMP protocol don't let ICMP ECHO Replies into your protected network. This would make your requests useless so you better block them.

7.2.5 ICMP Time to Live Exceeded in Transit (Type 11 Code 0)
To eliminate traceroute and Reverse Mapping techniques we do not want to let a Time-to-Live Exceeded code 0 messages go back to the malicious computer attacker.

7.2.6 ICMP Fragmentation Reassembly Time Exceeded (Type 11 Code 1)
By blocking this ICMP type we eliminate the usage of a Host Detection technique, which sends only few fragments, form a fragmented datagram, and force the probed host to send us an ICMP Fragmentation Reassembly Time Exceeded error message back revealing his existence.

7.2.7 ICMP Parameter Problem
We have demonstrated that host detection can be made with bad IP Header packets, which would elicit various ICMP Parameter Problem and ICMP Destination Unreachable error messages from the probed machines.

7.2.8 ICMP Time Stamp Request & Reply
Time Stamp requests & replies can be used for Host Detection and Inverse Mapping.

7.2.9 ICMP Address Mask Request and Reply
Address Mask request & reply can be used for host detection and Inverse Mapping.

7.2.10 The liability Question
System administrator / Network administrator don't want to be held liable for an attack generated from there network by an abusive user (or a malicious computer attacker using a compromised system within the network). Therefore blocking some types of ICMP traffic from the protected network to the outside world is recommended for liability reasons:

- Destination Unreachable Codes 2-4
    - ICMP Destination Unreachable error messages 2-4 ("Port Unreachable", "Protocol Unreachable" and "Fragmentation Needed and DF Flag was Set") is a group of messages that are hard error conditions and when received should terminate a connection.

      This allow an attacker to send *fake* Destination Unreachable codes 2-4 to terminate valid connections between the attacked target and other hosts on the void.

      Old TCP/IP implementations terminat TCP connections when receiving those error messages. Modern TCP/IP implementations no longer terminate a TCP connection when receiving those error messages
- Source Quench messages
    - Since hosts still react to Source Quenches by slowing communication, they can be used as a Denial-of-Service measure.
- Redirect messages
    - If you can forge ICMP Redirect packets, and if your target host pays attention to them - ICMP Redirects may be employed for denial of service attacks, where a

host is sent a route that loses it connectivity, or is sent an ICMP Network Unreachable packet telling it that it can no longer access a particular network.

This means that all outbound ICMP traffic should be disallowed.

7.3 Other Considerations

If you want to maintain strong ICMP filtering rules with your Firewall/Filtering-Device I suggest you block all incoming ICMP traffic except for Type 3 Code 4, which is used by the Path MTU Discovery process[50]. ICMP Type 3 Code 4 should be allowed from the Internet to your DMZ at least. Opening your Internal segmentation to this kind of traffic is questionable and depends on the facilities / activities / usage of the site and the level of filtering you wish to maintain.

If you will block incoming ICMP "Fragmentation Needed and Don't Fragment Bit was Set" your network performance will suffer from degradation. You should understand the security risks involving in opening this kind of traffic to your DMZ (& protected network) - The possibility of a Denial-of-Service, Inverse Mapping, Host Detection, and a one-way Covert communication channel (which was not been seen in the wild yet).

Another consideration could be the usage of network troubleshooting tools such as traceroute and ping. In the case of traceroute if the filtering device you are using does not support Stateful inspection with ICMP than allowing ICMP TTL Exceeded In Transit (Type 11, code 0) error messages inside the protected network could lead to various security hazards. The same goes with ping, where ICMP ECHO reply is even more dangerous when allowed inside the protected network (Inverse Mapping, Covert Channel and more security risks).

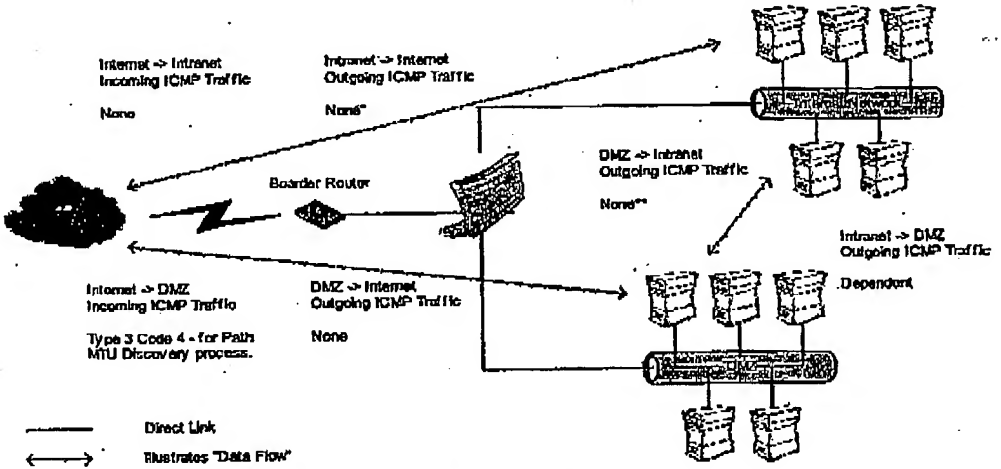
You can limit the number of systems that need to use the network troubleshooting tools with ACL, but bear in mind that those systems could be mapped from the Internet – and this is only the tip of the Iceberg.

Internal Host(s) performance considerations – When blocking incoming ICMP Destination Unreachable Network/Host/Protocol/Port Unreachable ICMP error messages coming from the Internet, host(s) would hang when the destination system's network is unreachable/when a host is unreachable/when a protocol on the destination machine is not available/a port on a destination machine is closed. They all would hang until the timeout counter would reach zero. This little Inconveniently is better than having the dangers other types of ICMP error messages inside your network can introduce.

Unless your filtering device is a real intelligence one, doing his work with dynamic tables and correlating correctly the ICMP replies with the requests, do not open your internal network segment to no ICMP traffic type.

Some might offer to use a Proxy server with the ICMP protocol between the Internet and you protected network(s). A Proxy Server is only a tunnel – remember that.

[50] See Appendix B: "Fragmentation Needed but the Don't Fragment Bit was set" and the Path MTU Discovery Process.



* You can have a dedicated Management station that would be allowed to use ICMP for troubleshooting purposes only. The various ICMP replies should be allowed only by a statful Inspection / Dynamic firewall. This means that no incoming ICMP is traffic is allowed to the management station, unless its correlated with a previous ICMP query this machine produced.

** If a malicious computer attacker breaks into the DMZ you do not want to provide him the means to scan internal machines & and the ability to query them directly.

Figure 14: Firewall ICMP Filtering Rules

7.4 Other Problems – Why it is important to filter ICMP traffic in the Internal segmentation

Consider the following realistic scenario:

You have an Internal segment built with Microsoft based operating system machines (for the sake of the example only). A malicious computer attacker might send you a Trojan that will have Host detection and/or mapping capabilities. It will be hidden in an Email message (either as attachment or some other thing) a naïve user will open. After activation it will start to map internal hosts and internal segments and send the information back to the malicious computer attacker.

What will be the easiest method in order to map internal host(s)? Ping them.

How many of you reading this research have "management segments" that are allowed to use the Ping utility in order to verify that some Hosts are alive?

If something like this Trojan gets its way to this segment than probably your entire internal networking infrastructure (or the important part of it) will be revealed.

Some one might think that strong filtering or a good anti virus might help – forget it. I have seen people separating their work environment to more than two or three computers, but they always use the Email, and need to surf the web... (good ways to send the collected Information out).

My suggestion is to configure internal host(s) not to answer for ICMP Query message types they should not answer for. I would restrict this to the maximum and not allow internal hosts to be queried with any ICMP Query message type.

Back to our monitoring problem - If you need to maintain management/monitoring capabilities, than I would suggest filtering the traffic in both ways from the management stations to the monitored systems in a way it would not be possible to simply query the last (dynamic filtering / stateful filtering with ICMP). Use a dedicated system for the querying and block the other machines in the management segment from doing so.

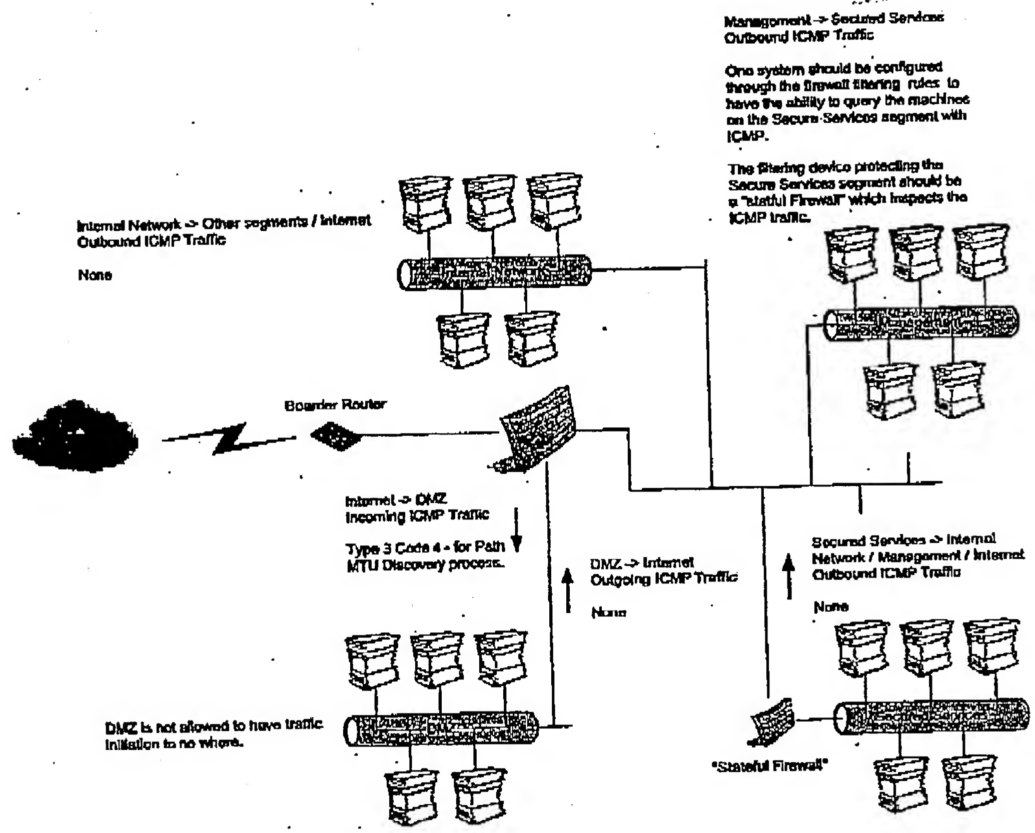

Management -> Secured Services
Outbound ICMP Traffic
One system should be configured through the firewall filtering rules to have the ability to query the machines on the Secure Services segment with ICMP.
The filtering device protecting the Secure Services segment should be a "stateful Firewall" which inspects the ICMP traffic.
Internal Network -> Other segments / Internet
Outbound ICMP Traffic
None
Boarder Router
Internet -> DMZ
Incoming ICMP Traffic
Type 3 Code 4 - for Path MTU Discovery process.
DMZ -> Internet
Outgoing ICMP Traffic
None
Secured Services -> Internal Network / Management / Internet
Outbound ICMP Traffic
None
DMZ is not allowed to have traffic initiation to no where.
"Stateful Firewall"

Figure 15: Internal segmentation ICMP Filtering Example

7.5 The Firewall

It is extremely important to block traffic, which is aimed at the Firewall itself. This rule will not block every thing. For example, ICMP error messages the firewall generates for various stimulus.

Some firewalls will hold a certain portion of a fragmented packet until the IP Header and the underlying protocol's header arrives. The ICMP error message for Fragment Reassembly Time Exceeded will not be of the Host, it will be of the Firewall spoofing it. Some Firewalls has the ability to spoof ICMP Echo Replies for Hosts they are defending. We will have the opportunity to fingerprint the operating system, which the firewall software is installed on.

We will gain an extremely important ability. Therefore it is recommended to have two basic rules when you configure your firewall's rule base. The first is to deny any traffic destined to the firewall and the second would be to deny any error messages (or other conditions such as TCP reject etc.) that might help a malicious computer attacker in his task to fingerprint the Firewall itself.

8.0 Conclusion

The ICMP protocol is a very powerful tool in the hands of smart malicious computer attackers. Mapping, detecting, and fingerprinting of hosts and networking devices can be done in various ways as I have outlined in this paper.

It is extremely important to understand that ICMP traffic can be used for other malicious activities other than scanning, such as:

- Denial of Service Attacks
- Distributed Denial of Service Attacks
- Covert Channel Communications

Therefore filtering Inbound and Outbound ICMP traffic is very important and may help you in preventing risks to your computing environment.

9.0 Acknowledgment

9.1 Acknowledgment for version 1.0

I would like to thank the following people for their help with/during this research.

Ariel Pisetsky for going over this paper correcting my English, and for his moral support.

Christopher Tresco, Systems Administrator at the Massachusetts Institute of Technology provided necessary test systems to verify my findings.

Special thanks to mr2940 for his patience while I introduced my new ideas.

James Cudney, Michael, Pat, for their support when the times where bad.

9.1 Acknowledgment for version 2.0

I would like to thank Alfredo Andres Omella author of SING for his help.

I would like to thank Fyodor for his help providing me with necessary test systems.

I would like to thank the people who provided feedback to the first version of this research paper, and to the people who provided feedback to my Bugtraq posts.

9.2 Acknowledgment for version 2.5

I would like to thank Alfredo Andres Omella author of SING, for implementing some of the ideas I had into his tool.

I would like to thank Fyodor for his help providing me with necessary test systems.

Christopher Tresco, Systems Administrator at the Massachusetts Institute of Technology provided necessary test systems to verify my findings.

I would like to thank Simple Nomad for his support.

I would like to thank the huge amount of people who provided feedback for my work.

Appendix A: The ICMP Protocol[51]

Internet Control Message Protocol (ICMP) is used for two types of operations: when a *router* or a *destination host* need to inform the source host about errors in a datagram processing, and for probing the network with request messages in order to determine general characteristics about the network (getting the information back, hopefully, with the reply messages).

Some of ICMP's characteristics are:

- ICMP uses IP as if it were a higher-level protocol, however, ICMP is already an internal part of IP, and must be implemented by every IP module.
- ICMP is used to provide feedback about some errors in a datagram processing, not to make IP reliable. Datagrams may still be undelivered without any report of their loss. If a higher level protocol that use IP need reliability he must implement it.
- No ICMP messages are sent in response to ICMP messages to avoid infinite repetitions. The exception is a response to ICMP query messages (ICMP Types 0,8-10,13-18. See Table 1 ICMP Query Messages).
- For fragmented IP datagrams ICMP messages are only sent about errors on fragment zero (first fragment).
- ICMP error messages are never sent in response to a datagram that is *destined* to a *broadcast* or a *multicast* address.
- ICMP error messages are never sent in response to a datagram sent as a link layer broadcast.
- ICMP error messages are never sent in response to a datagram whose source address does not represents a unique host – the source IP address cannot be *zero*, a *loopback* address, a *broadcast* address or a *multicast* address.
- ICMP Error messages are never sent in response to an IGMP massage of any kind.
- When an ICMP message of ***unknown type*** is received, it must be silently *discarded*.
- Routers will almost always generate ICMP messages but when it comes to a destination host(s), the number of ICMP messages generated is implementation dependent.

ICMP Query Messages	ICMP error Messages
Echo Router Advertisement Router Solicitation Time Stamp Information Address Mask	Destination Unreachable Source Quench Redirect Time Exceeded Parameter Problem

Table 16: ICMP message types

[51] ICMP is described in RFC 972 (http://www.ietf.org/rfc/rfc0972.txt) with updates in: RFC 896 (Source Quench), RFC 950 (Address Mask Extensions). RFC 1191 (Path MTU Discovery) & RFC 1256 (Router Discovery). Further clarifications about the ICMP protocol are included in RFC 1122 and in RFC 1812. STD 2 has redefine and clarified much of ICMP's core functionality.

A.1 ICMP Messages

ICMP messages are sent in IP datagrams. The protocol number will be always one (ICMP), and the Type-of-Service will be zero. The IP data field will contain the actual ICMP message:

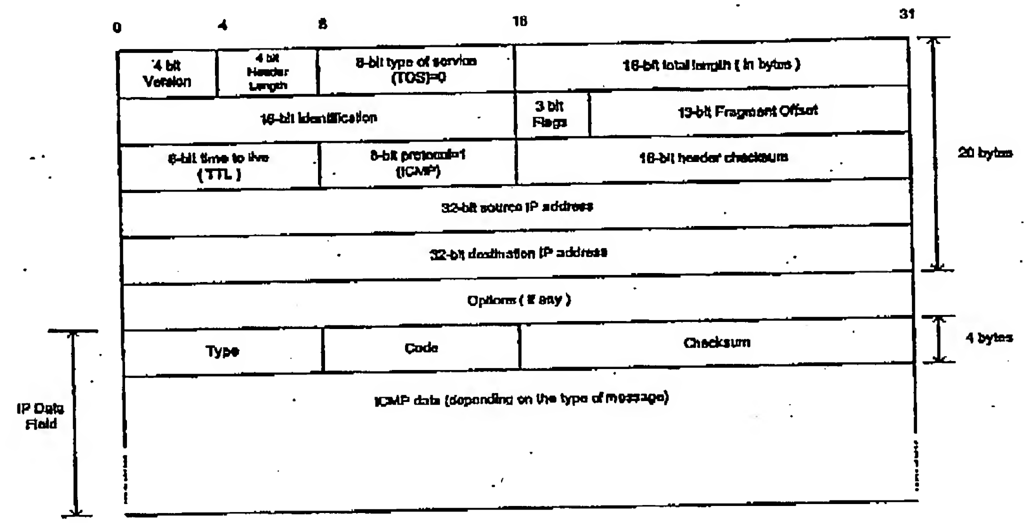


Figure 16: ICMP Message Format

ICMP error message length

Every ICMP error message includes the Internet (IP) Header and *at least* the first 8 data octets (bytes) of the datagram that triggered the error; more than 8 octets (bytes) *may* be sent; this header and data must be unchanged from the received datagram.

The **TYPE** field specifies the type of the message, while the error code for the datagram reported on by this ICMP message is contained in the **CODE** field. The code interpretation is dependent upon the message type.

Type	Name	Code
0	Echo Reply	0 No Code
1	Unassigned	
2	Unassigned	
3	Destination Unreachable[52]	
		0 Net Unreachable
		1 Host Unreachable
		2 Protocol Unreachable
		3 Port Unreachable
		4 Fragmentation Needed and Don't Fragment was Set
		5 Source Route Failed
		6 Destination Network Unknown
		7 Destination Host Unknown
		8 Source Host Isolated[53]
		9 Communication with Destination Network is Administratively Prohibited[54]
		10 Communication with Destination Host is Administratively Prohibited[55]
		11 Destination Network Unreachable for Type of Service.
		12 Destination Host Unreachable for Type of Service.
		13 Communication Administratively Prohibited.
		14 Host Precedence Violation
		15 Precedence cutoff in effect
4	Source Quench	0 No Code
5	Redirect	
		0 Redirect Datagram for the Network (or subnet)
		1 Redirect Datagram for the Host
		2 Redirect Datagram for the Type of Service and Network
		3 Redirect Datagram for the Type of Service and Host
6	Alternate Host Address	0 Alternate Address for Host
7	Unassigned	
8	Echo Request	0 No Code
9	Router Advertisement	0 No Code
10	Router Selection	0 No Code
11	Time Exceeded	
		0 Time to Live exceeded in Transit
		1 Fragment Reassembly Time Exceeded
12	Parameter Problem	
		0 Pointer indicates the error
		1 Missing a Required Option
		2 Bad Length
13	Timestamp	0 No Code
14	Timestamp Reply	0 No Code

[52] RFC 972 defines codes 1-5. RFC 1122 defines codes 6-12. RFC 1812 defines codes 13-15.

[53] Reserved for use by U.S. military agencies.

[54] Reserved for use by U.S. military agencies.

[55] Reserved for use by U.S. military agencies.

Type	Name	Code
15	Information Request	0 No Code
16	Information Reply	0 No Code
17	Address Mask Request	0 No Code
18	Address Mask Reply	0 No Code
19	Reserved (for Security)	0 No Code
	20-29 reserved (for Robustness Experiment)	
30	Traceroute	
31	Datagram Conversion Error	
32	Mobile Host Redirect	
33	IPv6 Where-Are-You	
34	IPv6 I-Am-Here	
35	Mobile Registration Request	
36	Mobile Registration Reply	
39	SKIP	
40	Photuris	0 Reserved 1 unknown security parameters index 2 valid security parameters, but authentication failed 3 valid security parameters, but decryption failed

Table 17: ICMP Types & Codes

Checksum – contains the 16bit one's complement of the one's complement sum of the ICMP message starting with the ICMP Type field. For computing this checksum, the checksum field is assumed to be zero.

Data

- With ICMP error messages it will contain a part of the original IP message for which this ICMP message was generated. The length of the DATA field equals the IP datagram length less the IP header length. Every ICMP error message includes the Internet (IP) Header and *at least* the first 8 data octets (bytes) of the datagram that triggered the error; more than 8 octets (bytes) *may* be sent; this header and data must be unchanged from the received datagram.
- With ICMP query messages the Data field will contain dependent information upon the query type.

A.1 ICMP Error Messages

ICMP error messages are used to report a problem that prevented delivery. The nature of the problem should be a non-transient delivery problem.

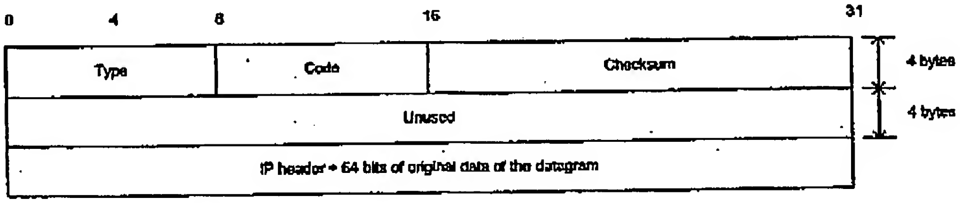


Figure 17: ICMP Error Message General Format

ICMP error message length

Every ICMP error message includes the IP Header (20 to 60 bytes) and *at least* the first 8 data bytes of the datagram that triggered the error; more than 8 bytes *may* be sent; this header and data must be unchanged from the received datagram.

AN ICMP error message length should be between 36 to 72 bytes.

The ICMP Protocol Rules for ICMP Error Messages

- ICMP Error messages are not sent for another ICMP Error message to prevent infinite loops.
- ICMP error messages are never sent in response to a datagram that is *destined* to a *broadcast* or a *multicast* address.
- ICMP error messages are never sent in response to a datagram sent as a link layer broadcast.
- ICMP error messages are never sent in response to a datagram whose source address does not represents a unique host – the source IP address cannot be *zero*, a *loopback* address, a *broadcast* address or a *multicast* address.
- ICMP Error messages are never sent in response to an IGMP massage of any kind.

A.1.1 ICMP Error Messages[56]

- Destination Unreachable (Type 3)
- Source Quench (Type 4)
- Redirect (Type 5)
- Time Exceeded (Type 11)
- Parameter Problem (Type 12)

A.1.1.1 Destination Unreachable (Type 3)

ICMP Destination Unreachable message type issued by a Destination Host:

A *destination host* issues a destination unreachable message when the protocol specified in the *protocol number* field of the original datagram is not active on the destination host, or the *specified port* is inactive.

ICMP Destination Unreachable message type issued by a Router:

[56] Some of the wording in this section are direct quotes from RFC 792 available from http://www.ietf.org/rfc/rfc0972.txt.

A *router* issue a destination unreachable message in response to a packet that it cannot forward because the destination (or next hop) is unreachable or a service is unavailable.

Code	Meaning	Explanation
0	Network Unreachable	Generated by a router if a route to the destination network is not available.
1	Host Unreachable	Generated by a router if a route to the destination host on a directly connected network is not available (does not respond to ARP).
2	Protocol Unreachable	Generated if the transport protocol designated in a datagram is not supported in the transport layer of the final destination.
3	Port Unreachable	Generated if the designated transport protocol (e.g. UDP) is unable to demultiplex the datagram in the transport layer of the final destination but has no protocol mechanism to inform the sender.
4	Fragmentation needed and DF flag Set	Generated if a router needs to fragment but cannot since the DF flag is set.
5	Source Route Failed	Generated if a router cannot forward a packet to the next hop in a source route option.
6	Destination Network Unknown	According to RFC 1812 this code should not be generated since it would imply on the part of the router that the destination network does not exist (net unreachable code 0 should be used instead of code 6).
7	Destination Host Unknown	Generated only when a router can determine (from link layer advice) that the destination host does not exist.
8	Source Host Isolated	Generated by a Router if it have been configured not to forward packets from source.
9	Communication with Destination Network is Administratively Prohibited	Generated by a Router if it has been configured to block access to the desired destination network.
10	Communication with Destination Host is Administratively Prohibited	Generated by a Router if it has been configured to block access to the desired destination host.
11	Network Unreachable for Type of Service	Generated by a router if a route to the destination network with the requested or default TOS is not available.
12	Host Unreachable for Type of Service	Generated if a router cannot forward a packet because its route(s) to the destination do not match either the TOS requested in the datagram or the default TOS (0).
13*	Communication Administratively Prohibited	Generated if a router cannot forward a packet due to administrative filtering (ICMP sender is not available at this time).
14	Host Precedence Violation	Sent by the first hop router to a host to indicate that a requested precedence is not permitted for the particular combination of source/destination host or network, upper layer protocol, and source/destination port.

Code	Meaning	Explanation
15	Precedence cutoff in effect	The network operators have imposed a minimum level of precedence required for operation, the datagram was sent with precedence below this level.

* Routers *may* have a configuration option that causes code 13 messages not to be generated. When this option is enabled, no ICMP error message is sent in response to a packet that is dropped because it's forwarding is administratively prohibited. Same is with type 14 & 15.

Table 18: Destination Unreachable Codes (Router)

The only type of ICMP Destination Unreachable error message, which is slightly different from the other, is Type 3 Code 4 – Fragmentation Needed but the Don't Fragment Bit was set.

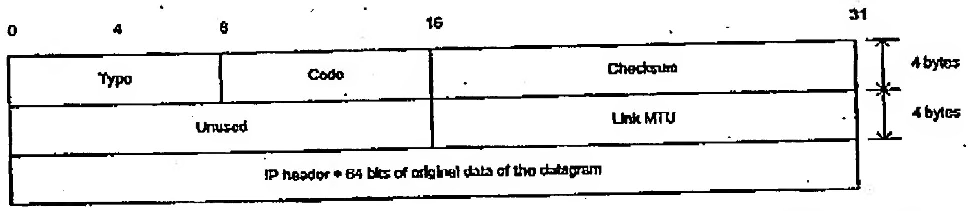


Figure 18: ICMP Fragmentation Needed but the Don't Fragment Bit was set Message Format

The Unused field will be 16 bits in length, instead of 32 bits, with this type of message. The rest of the 16 bits will be used to carry the MTU used for the link that could not deliver the datagram to the next hop (or destination) because the size of the datagram was too big to carry. Since this datagram could not be fragmented (the DF Bit was set) an error message has been sent to the sender indicating that a lower MTU should be used, hinting the size of the next hops links.

A.1.1.2 Source Quench (Type 4)

ICMP Source Quench message type issued by a Router:
If a *router* sends this message, it means that the router does not have the buffer space needed to queue the datagrams for output to the next network on the route to the destination network.

RFC 1812 specify that a router *should not* generate Source Quench messages, but a router that does originate Source Quench message *must* be able to limit the rate at which they are generated (because it consumes bandwidth and it is an ineffective antidote to congestion).

A router receiving an ICMP Source Quench message type:
When a router receives such a message it *may* ignore it.

ICMP Source Quench message type issued by a Host:
If a *destination host* sends this message (it *may* be implemented), it means that the datagrams arrive too fast to be processed. The ICMP source quench message is a request to the host to cut back the rate, which it is sending traffic to the Internet destination.

The ICMP header code would be always zero.

Host receiving an ICMP Source Quench message type:
An ICMP Source Quench message *must* be reported to the transport layer, UDP or TCP, the host should throttle itself back for a period of time, than gradually increase the transmission rate again.

A.1.1.3 Redirect (Type 5)

ICMP Redirect message type issued by a Router:
If a router generates this message, it means the host should send future datagrams for the network to the router who's IP is given in the ICMP message. The router should be always on the same subnet as the host who sent the datagram and the router that generated the ICMP redirect message.

A routing loop is generated when the router IP address matches the source IP address in the original datagram header.

Routers *must not* generate a Redirect Message unless *all* the following conditions are met:

- The packet is being forwarded out the same physical interface that it was received from,
- The IP source address in the packet is on the same Logical IP (Sub) network as the next-hop IP address, and
- The packet does not contain an IP source route option.

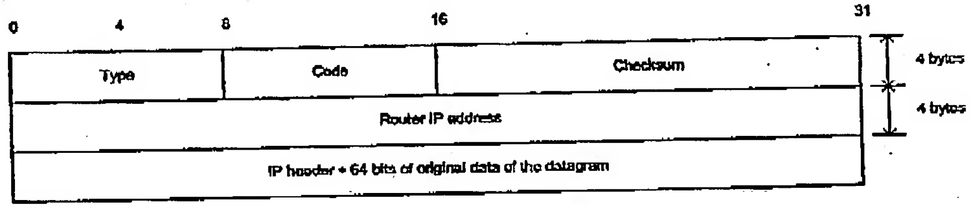


Figure 19: ICMP Redirect Message Format

A router receiving an ICMP Redirect message type:
A router *may* ignore ICMP Redirects when choosing a path for a packet originated by the router if the router is running a routing protocol or if forwarding is enabled on the router and on the interface over which the packet is being sent.

Four different codes can appear in the code field:

Code	Meaning
0	Redirect Datagram for the Network (or subnet)
1	Redirect Datagram for the Host

Code	Meaning
2	Redirect Datagram for the Type of Service and Network
3	Redirect Datagram for the Type of Service and Host

Table 19: Redirect Codes

ICMP Redirect message type issued by a Host:
A *host should not* send an ICMP Redirect message. Redirects are to be sent only by routers.

Host receiving an ICMP Redirect message type:
A *host* receiving a Redirect message *must* update its routing information accordingly. Every host *must* be prepared to accept both Host and Network Redirects.

The Redirect message *should* be silently discarded with the following cases:

- The new gateway address it specifies is not on the same connected (sub-) net through which the Redirect arrived.
- If the source of the Redirect is not the current first-hop gateway for the specified destination.

A.1.1.4 Time Exceeded (Type 11)

ICMP Time Exceeded message type issued by a Router:
If a *router* discovers that the Time-To-Live field in an IP header of a datagram he process equals zero he will discard the datagram and generate an ICMP Time Exceeded Code 0 – transit TTL expired (this can also be an indicator of a routing loop problem).

When the router reassemble a packet that is destined for the router, it is acting as an Internet host. Host rules apply also when the router *receives* a Time Exceeded message.

A router *must* generate an ICMP Time Exceeded message code 0 when it discards a packet due to an expired TTL field. A router *may* have a per-interface option to disable origination of these messages on that interface, but that option *must* default to allowing the messages to be originated.

ICMP Time Exceeded message type issued by a Host:
If a *host* cannot reassemble a fragmented datagram due to missing fragments within its time limit it will discard the datagram and generate an ICMP Time Exceeded Code 1 – reassembly TTL Exceeded.

A.1.1.5 Parameter Problem (Type 12)

ICMP Parameter Problem message is sent when a router (*must* generate this message) or a host (*should* generate this message) process a datagram and finds a problem with the IP header parameters. It is only sent if the error caused the datagram to be discarded.

The Parameter Problem message is generated usually for any error not specifically covered by another ICMP message.

If code 0 is used, the pointer field will point to the exact byte in the original IP Header, which caused the problem.

Three different codes can appear in the code field:

Codes	Meaning
0	Pointer indicates the error (unspecified error)
1	Missing a Required Option
2	Bad Length

Table 20: Parameter Problem Codes

Receipt of a parameter problem message generally indicates some local or remote implementation error.

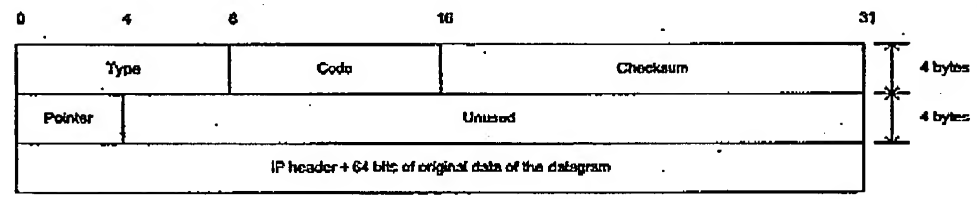


Figure 20: ICMP Parameter Problem Message Format

Appendix B: ICMP "Fragmentation Needed but the Don't Fragment Bit was set" and the Path MTU Discovery Process [57]

When one host needs to send data to another host, the data is transmitted in a series of IP datagrams. We wish the datagrams be the largest size possible that does not require fragmentation[58] along the path from the source host to the destination host.

Fragmentation by the IP layer raises few problems:

- If one fragment from a packet is dropped, we need to retransmit the whole packet.
- Load on the routers, which needs to do the fragmentation.
- Some simpler firewalls would block all fragments because they do not contain the header information for a higher layer protocol needed for filtering.

The Maximum Transfer Unit (***MTU***) is a link layer restriction on the maximum number of bytes of data in a single transmission. The smallest MTU of any link on the current path between two hosts is called the ***Path MTU***.

B.1 The PATH MTU Discovery Process

We use the Don't Fragment Bit Flag in the IP header to dynamically discover the Path MTU of a given route: The source host assumes that the PMTU of a path is the known MTU of its first hop. He will send all datagrams with that size, and set the Don't Fragment Bit. If along the path to the destination host, there is a router that needs to fragment the datagram in order to pass it to the next hop, an ICMP error message (Type 3 Code 4 "Fragmentation Needed and DF set") will be generated, since the Don't Fragment bit was set. When the sending host receives the ICMP error message he should reduce his assumed PMTU for the path.

The process can end when the estimated PMTU is low enough for the datagrams not to be fragmented. The source host itself can stop the process if he is willing to have the datagrams fragmented in some circumstances.

Usually the DF bit would be set in all datagrams, so if a route changes to the destination host, and the PMTU is lowered, than we would discover it.

The PMTU of a path might be increased over time, again because of a change in the routing topology. To detect it, a host should periodically increase its assumed PMTU for that link.

The link MTU field in the ICMP "Fragmentation Needed and DF set" error message, carries the MTU of the constricting hop, enabling the source host to know the exact value he needs to set the PMTU for that path to allow the voyage of the datagrams beyond that point (router) without fragmentation.

B.2 Host specification

A host must reduce his estimated PMTU for the relevant path when he receives the ICMP "Fragmentation Needed and the DF bit was set" error message. RFC 1191 does not outline a specific behavior that is expected from the sending host, because different applications may have different requirements, and different implementation architectures may favor different strategies.

[57] RFC 1191, http://www.ietf.org/rfc/rfc1191.txt, J. Mogul, S. Deering.

[58] When we send a packet that it is too large to be sent across a link as a single unit, a router needs to slice/split the packet into smaller parts, which contain enough information for the receiver to reassemble them. This is called fragmentation.

The only required behavior is that a host *must* attempt to avoid sending more messages with the same PMTU value in the near future. A host can either cease setting the Don't Fragment bit in the IP header (and allow fragmentation by the routers in the way) or reduce the datagram size. The better strategy would be to lower the message size because fragmentation will cause more traffic and consume more Internet resources.

A host using the PMTU Discovery process *must* detect decreases in Path MTU as fast as possible. A host *may* detect increases in Path MTU, by sending datagrams larger than the current estimated PMTU, which will usually be rejected by some router on the path to a destination since the PMTU usually will not increase. Since this would generate traffic back to the host, the check for the increases must be done at infrequent intervals. The RFC specify that an attempt for detecting an increasment *must not* be done less than 10 minutes after a datagram Too Big has been received for the given destination, or less than 2 minute after a previously successful attempt to increase.

The sending host must know how to handle an ICMP "Fragmentation Needed and the DF bit was set" error message that was sent by a device who does not know how to handle the PMTU protocol and does not include the next-hop MTU in the error message. Several strategies are available:

- The PMTU should be set to the minimum between the currently assumed PMTU and 576[59]. The DF bit should not be set in future datagrams for that path.
- Searching for the accurate value for the PMTU for a path. We keep sending datagrams with the DF bit set with lowered PMTU until we do not receive errors.

A host must not reduce the estimation of a Path MTU value below 68 bytes.

A host MUST not increase its estimate of the Path MTU in response to the contents of a Datagram Too Big message.

B.3 Router Specification

When a router cannot forward a datagram because it exceeded the MTU of the next-hop network and the Don't Fragment bit was set, he is required to generate an ICMP Destination Unreachable message to the source of the datagram., with the appropriate code indicating "Fragmentation needed and the Don't Fragment Bit was set". In the error message the router *must* include the MTU of the next-hop in a 16bit field inside the error message.

<table>
<tr><th>0</th><th>8</th><th>16</th><th>31</th></tr>
<tr><td>Type=3</td><td>Code=4</td><td colspan="2">Checksum</td></tr>
<tr><td colspan="2">Unused (zero)</td><td colspan="2">Link MTU</td></tr>
<tr><td colspan="4">IP header + 64 bits of original data of the datagram</td></tr>
</table>

Figure 21: ICMP Fragmentation Required with Link MTU

[59] The usage of the lesser between 576 and the first-hop MTU as the PMTU for a destination, which is not connected to the same network was the old implementation. The results were the use of smaller datagrams than necessary, waste of Internet resources, and not being optimal.

The value of the next-hop MTU field should be set to the size in bytes of the largest datagram that could be forwarded, along the path of the original datagram, without being fragmented by this router. The size includes IP header plus IP data and no lower level headers should be included.

Because every router should be able to forward a datagram of 68 bytes without fragmenting it, the link MTU field should not contain a value less than 68.

B.4 The TCP MSS (Maximum Segment Size) Option and PATH MTU Discovery Process

The RFC specify that a host that is doing Path MTU Discovery *must not* send datagrams larger than 576 bytes unless the receiving host grants him permission.

When we are establishing a TCP connection both sides announce the maximum amount of data in one packet that should be sent by the remote system – The maximum segment size, MSS (if one of the ends does not specify an MSS, it defaults to 536 – there is no permission from the other end to send more than this amount). The packet generated would be, normally, 40 bytes larger than the MSS; 20 bytes for the IP header and 20 bytes for the TCP header. Most systems announce an MSS that is determined from the MTU on the interface that the traffic to the remote system passes out from the system through.

Each side upon receiving the MSS of the other side should not send any segments larger than the MSS received, regardless of the PMTU. After receiving the MSS value the Path MTU Discovery process will start to take affect. We will send our IP packets with the DF bit set allowing us to recognize points in the path to our destination that cannot process packets larger as the MSS of the destination host plus 40 bytes. When such an ICMP error message arrives, we should lower the PMTU to a path (according to the link MTU field, or if not used, to use the rules regarding the old implementation) and retransmit. The value of the link MTU cannot be higher than the MSS of the destination host. When retransmission occurs resulting from ICMP type 3 code 4 error message, the congestion windows should not change, but slow start should be initiated. The process continues until we adjust the correct PMTU of a path (not receiving ICMP error messages from the intermediate routers) which will allow us to fragment at the TCP layer which is much more efficient than at the IP layer.

Appendix C: Mapping Operating Systems for answering/discarding ICMP query message types

Operating System	Info. Request	Time Stamp Request	Address Mask Request	Address Mask Request Frag.	IP TTL on ICMP datagrams - In Reply -	IP TTL on ICMP datagrams - In Req. -
Debian GNU/ LINUX 2.2, Kernel 2.4 test 2 (*)	-	+	-	-	255	64
Redhat LINUX 6.2 Kernel 2.2.14 (*)	-	+	-	-	255	64
LINUX Kernel 2.0.x					64	64
FreeBSD 4.0 (*)	-	+	-	-	255	255
FreeBSD 3.4	-	+	-	-	255	255
OpenBSD 2.7	-	+	-	-	255	255
OpenBSD 2.6	-	+	-	-	255	255
NetBSD	-	+	-	-	255	
BSDI BSD/OS 4.0	-	+	-	-	255	
BSDI BSD/OS 3.1	-	+	-	-	255	
Solaris 2.5.1	-	+	+	+ (0.0.0.0)	255	255
Solaris 2.6	-	+	+	+ (0.0.0.0)	255	255
Solaris 2.7 (*)	-	+	+	+ (0.0.0.0)	255	255
Solaris 2.8	-	+	+	+ (0.0.0.0)	255	255
HP-UX v10.20	+	+	-	-	255	255
HP-UX v11.0	-	-	+	+ (0.0.0.0)	255	
Compaq Tru64 v5.0 (*)	+	+	-	-	64	
Irix 6.5.3 (*)	-	+	-	-	255	
Irix 6.5.8 (*)	-	+	-	-	255	
AIX 4.1 (*)	+	+	-	-	255	
AIX 3.2 (*)	+	+	-	-	255	
ULTRIX 4.2 – 4.5 (*)	+	+	+	+	255	
OpenVMS v7.1-2 (*)	+	+	+	+	255	
Novell Netware 5.1 SP1 (*)	-	-	-	-	128	
Novell Netware 5.0 (*)	-	-	-	-	128	
Novell Netware 3.12	-	-	-	-	128	
Windows 95	-	-	+	+	32	32
Windows 98 (*)	-	+	+	+	128	32
Windows 98 SE (*)	-	+	+	+	128	32
Windows ME (*)	-	+	-	-	128	32
Windows NT 4 WRKS SP 3 (*)	-	-	+	+	128	32
Windows NT 4 WRKS SP 6a (*)	-	-	-	-	128	32
Windows NT 4 Server SP4	-	-	-	-	128	32
Windows 2000 Professional (*)	-	+	-	-	128	128
Windows 2000 Server (*)	-	+	-	-	128	128

Networking Devices	Info. Request	Time Stamp Request	Address Mask Request	Address Mask Request Frag.	IP TTL on ICMP datagrams - In Reply -	IP TTL on ICMP datagrams - In Req. -
Cisco Catalyst 5505 with OSS v4.5	+	+	+	-	60	60
Cisco Catalyst 2900XL with IOS 11.2	+	+	-	-	255	
Cisco 3600 with IOS 11.2	+	+	-	-	255	
Cisco 7200 with IOS 11.3	+	+	-	-	255	255
Intel Express 8100 ISDN Router (*)	-	-	+	+	64	

Appendix D: ICMP Query Message Types with Code field I=0

Operating System	Info. Request	Time Stamp Request	Address Mask Request		ECHO Request	
Debian GNU/ LINUX 2.2, Kernel 2.4 test 2 (*)	-	+ (0)	-		+ (I=0)	
Redhat LINUX 6.2 Kernel 2.2.14 (*)	-	+ (0)	-		+ (I=0)	
FreeBSD 4.0 (*)	-	+ (I=0)	-		+ (I=0)	
FreeBSD 3.4	-	+ (I=0)	-			
OpenBSD 2.7	-	+ (I=0)	-		+ (I=0)	
OpenBSD 2.6	-	+ (I=0)	-		+ (I=0)	
NetBSD	-	+ (I=0)	-		+ (I=0)	
BSDI BSD/OS 4.0 (*)	-	+ (I=0)	-		+ (I=0)	
BSDI BSD/OS 3.1 (*)	-	+ (I=0)	-		+ (I=0)	
Solaris 2.5.1	-	+ (I=0)	+ (I=0)		+ (I=0)	
Solaris 2.6	-	+ (I=0)	+ (I=0)		+ (I=0)	
Solaris 2.7 (*)	-	+ (I=0)	+ (I=0)		+ (I=0)	
Solaris 2.8	-	+ (I=0)	+ (I=0)		+ (I=0)	
HP-UX v10.20	+ (I=0)	+ (I=0)	-			
HP-UX v11.0	-	-	+ (I=0)		+ (I=0)	
Compaq Tru64 v5.0 (*)	+ (I=0)	+ (I=0)	-		+ (I=0)	
Irix 6.5.3 (*)	-	+ (I=0)	-		+ (I=0)	
Irix 6.5.8 (*)	-	+ (I=0)	-		+ (I=0)	
AIX 4.1 (*)	+ (I=0)	+ (I=0)	-		+ (I=0)	
Aix 3.2 (*)	+ (I=0)	+ (I=0)	-			
ULTRIX 4.2 - 4.5 (*)	+ (I=0)	+ (I=0)	+ (I=0)		+ (I=0)	
OpenVMS v7.1-2 (*)	+ (I=0)	+ (I=0)	+ (I=0)		+ (I=0)	
Novell Netware 5.1 SP1 (*)	-	-	-		+ (I=0)	
Novell Netware 5.0 (*)	-	-	-		+ (I=0)	
Novell Netware 3.12 (*)	-	-	-		+ (I=0)	
Windows 95	-	-	+		+ (0)	
Windows 98 (*)	-	- (CHANGE)	+		+ (0)	
Windows 98 SE (*)	-	- (CHANGE)	+		+ (0)	
Windows ME (*)	-	- (CHANGE)	-		+ (0)	
Windows NT 4 WRKS SP 3 (*)	-	-	+		+ (0)	
Windows NT 4 WRKS SP 6a (*)	-	-	-		+ (0)	
Windows NT 4 Server SP4	-	-	-		+ (0)	
Windows 2000 Professional (*)	-	- (CHANGE)	-		+ (0)	
Windows 2000 Server (*)	-	- (CHANGE)	-		+ (0)	

Networking Devices	Info. Request	Time Stamp Request	Address Mask Request		ECHO Request	
Cisco Catalyst 5505 with OSS v4.5	+	+	+		+ (10)	
Cisco Catalyst 2900XL with IOS 11.2	+	+	-		+ (10)	
Cisco 3600 with IOS 11.2					+ (10)	
Cisco 7200 with IOS 11.3	+	+	-		+ (10)	
Intel Express 8100 ISDN Router (*)						

Appendix E: ICMP Query Message Types aimed at a Broadcast Address

Operating System	Info. Request	Time Stamp Request	Address Mask Request		Echo Request	
	Broadcast	Broadcast	Broadcast		Broadcast	
Debian GNU/ LINUX 2.2, Kernel 2.4 test 2					+	
Redhat LINUX 6.2 Kernel 2.2.14 (*)	-	+	-		+	
FreeBSD 4.0 (*)	-	-	-		-	
FreeBSD 3.4	-	-	-		-	
OpenBSD 2.7	-	-	-		-	
OpenBSD 2.6	-	-	-		-	
NetBSD						
BSDI BSD/OS 4.0 (*)						
BSDI BSD/OS 3.1 (*)						
Solaris 2.5.1	-	+	-		+	
Solaris 2.6	-	+	-		+	
Solaris 2.7	-	+	-		+	
Solaris 2.8	-	+	-		+	
HP-UX v10.20	+	+	-		+	
Compaq Tru64 v5.0 (*)						
Irix 6.5.3 (*)						
Irix 6.5.8 (*)						
AIX 4.1 (*)						
AIX 3.2 (*)						
ULTRIX 4.2 – 4.5 (*)						
OpenVMS v7.1-2 (*)						
Novell Netware 5.1 SP1 (*)						
Novell Netware 5.0 (*)						
Novell Netware 3.12 (*)						
Windows 95						
Windows 98	-	-	-		-	
Windows 98 SE (*)	-	-	-		-	
Windows ME (*)	-	-	-		-	
Windows NT 4 WRKS SP 3 (*)	-	-	-		-	
Windows NT 4 WRKS SP 6a (*)	-	-	-		-	
Windows NT 4 Server SP4	-	-	-		-	
Windows 2000 Professional (*)	-	-	-		-	
Windows 2000 Server (*)	-	-	-		-	

Networking Devices	Info. Request Broadcast	Time Stamp Request Broadcast	Address Mask Request Broadcast		Echo Broadcast	
Cisco Catalyst 5505 with OSS v4.5	+	+	+		+	
Cisco Catalyst 2900XL with IOS 11.2	+	-	-		+	
Cisco 3600 with IOS 11.2	+	-	-			
Cisco 7200 with IOS 11.3	+	-	-		+	
Intel Express 8100 ISDN Router (*)	-	-	-		-	Big Question Marks

Appendix F: Precedence Bits Echoing with ICMP Query Request & Reply

Operating System	Information Request With Precedence!=0	Time Stamp Request With Precedence!=0	Address Mask Request With Precedence!=0	Echo Request With Precedence!=0
Debian GNU/ LINUX 2.2, Kernel 2.4 test 2	Not Answering	!=0x00	Not Answering	!=0x00
Redhat LINUX 6.2 Kernel 2.2.14	Not Answering	!=0x00	Not Answering	!=0x00
FreeBSD 4.0	Not Answering	!=0x00	Not Answering	!=0x00
FreeBSD 4.1.1	Not Answering	!=0x00	Not Answering	
OpenBSD 2.7	Not Answering		Not Answering	!=0x00
OpenBSD 2.6	Not Answering		Not Answering	!=0x00
NetBSD	Not Answering		Not Answering	!=0x00
BSDI BSD/OS 4.0	Not Answering		Not Answering	!=0x00
BSDI BSD/OS 3.1	Not Answering		Not Answering	!=0x00
Solaris 2.5.1	Not Implemented			
Solaris 2.6	Not Implemented	!=0x00	!=0x00	!=0x00
Solaris 2.7	Not Implemented	!=0x00	!=0x00	!=0x00
Solaris 2.8	Not Implemented	!=0x00	!=0x00	!=0x00
HP-UX v10.20			Not Answering	
HP-UX v11.0	Not Answering	Not Answering	!=0x00 -> 0x00	!=0x00 -> 0x00
Compaq Tru64 v5.0	!=0x00	!=0x00	Not Answering	!=0x00
AIX 4.3	!=0x00	!=0x00	Not Answering	!=0x00
AIX 4.2.1	!=0x00	!=0x00	Not Answering	!=0x00
AIX 4.1	!=0x00	!=0x00	Not Answering	!=0x00
AIX 3.2	!=0x00	!=0x00	Not Answering	!=0x00
ULTRIX 4.2 – 4.5	0x00	0x00	0x00	0x00
OpenVMS v7.1-2	0x00	0x00	0x00	!=0x00
Windows 95	Not Answering	Not Answering		
Windows 98	Not Answering			!=0x00
Windows 98 SE	Not Answering	0x00	0x00	!=0x00
Windows ME	Not Answering	0x00	Not Answering	!=0x00
Windows NT 4 WRKS SP 3	Not Answering	Not Answering		!=0x00
Windows NT 4 WRKS SP 6a	Not Answering	Not Answering	Not Answering	!=0x00
Windows NT 4 Server SP4	Not Answering	Not Answering	Not Answering	!=0x00
Windows 2000 Professional	Not Answering	0x00	Not Answering	0x00
Windows 2000 Server	Not Answering	0x00	Not Answering	0x00

Appendix G: ICMP Query Message Types with TOS! = 0

Operating System	Information Request With TOS!=0x00	Time Stamp Request With TOS!=0x00	Address Mask Request With TOS!=0x00	Echo Request With TOS!=0x00
Debian GNU/ LINUX 2.2, Kernel 2.4 test 2 (*)	Not Answering	!=0x00	Not Answering	!=0x00
Redhat LINUX 6.2 Kernel 2.2.14 (*)	Not Answering	!=0x00	Not Answering	!=0x00
FreeBSD 4.0 (*)	Not Answering	!=0x00	Not Answering	!=0x00
FreeBSD 3.4	Not Answering		Not Answering	
OpenBSD 2.7 (*)	Not Answering	!=0x00	Not Answering	!=0x00
OpenBSD 2.6	Not Answering	!=0x00	Not Answering	!=0x00
NetBSD	Not Answering	!=0x00	Not Answering	!=0x00
BSDI BSD/OS 4.0 (*)	Not Answering	!=0x00	Not Answering	!=0x00
BSDI BSD/OS 3.1 (*)	Not Answering	!=0x00	Not Answering	!=0x00
Solaris 2.5.1	Not Implemented			
Solaris 2.6	Not Implemented			
Solaris 2.7 (*)	Not Implemented	!=0x00	!=0x00	!=0x00
Solaris 2.8 (*)	Not Implemented	!=0x00	!=0x00	!=0x00
HP-UX v10.20			Not Answering	
HP-UX v11.0	Not Answering	Not Answering	!=0x00	!=0x00
Compaq Tru64 v5.0 (*)		!=0x00	Not Answering	!=0x00
Irix 6.5.3 (*)	Not Answering	!=0x00	Not Answering	!=0x00
Irix 6.5.8 (*)	Not Answering	!=0x00	Not Answering	!=0x00
AIX 4.1 (*)		!=0x00	Not Answering	!=0x00
AIX 3.2 (*)		!=0x00	Not Answering	!=0x00
ULTRIX 4.2 – 4.5 (*)		0x00	0x00	0x00
OpenVMS v7.1-2 (*)		!=0x00	!=0x00	!=0x00
Novell Netware 5.1 SP1 (*)	Not Answering	Not Answering	Not Answering	0x00
Novell Netware 5.0 (*)	Not Answering	Not Answering	Not Answering	0x00
Novell Netware 3.12 (*)	Not Answering	Not Answering	Not Answering	0x00
Windows 95	Not Answering	Not Answering		
Windows 98 (*)	Not Answering	0x00	0x00	!=0x00
Windows 98 SE (*)	Not Answering	0x00		!=0x00
Windows ME (*)	Not Answering	0x00	Not Answering	!=0x00
Windows NT 4 WRKS SP 3 (*)	Not Answering	Not Answering		!=0x00
Windows NT 4 WRKS SP 6a (*)	Not Answering	Not Answering	Not Answering	!=0x00
Windows NT 4 Server SP4	Not Answering	Not Answering	Not Answering	!=0x00
Windows 2000 Professional (*)	Not Answering	0x00	Not Answering	0x00
Windows 2000 Server (*)	Not Answering	0x00	Not Answering	0x00

Appendix H: Echoing the TOS Byte Unused bit

Operating System	Information Request With Unused=1	Time Stamp Request With Unused=1	Address Mask Request With Unused=1	Echo Request With Unused=1
Debian GNU/ LINUX 2.2, Kernel 2.4 test 2	Not Answering	0x1	Not Answering	0x1
Redhat LINUX 6.2 Kernel 2.2.14	Not Answering	0x1	Not Answering	0x1
FreeBSD 4.0	Not Answering	0x1	Not Answering	0x1
FreeBSD 4.1.1	Not Answering	0x1	Not Answering	0x1
OpenBSD 2.7	Not Answering		Not Answering	
OpenBSD 2.6	Not Answering		Not Answering	
NetBSD	Not Answering		Not Answering	
BSDI BSD/OS 4.0	Not Answering		Not Answering	
BSDI BSD/OS 3.1	Not Answering		Not Answering	
Solaris 2.5.1	Not Implemented			
Solaris 2.6	Not Implemented	0x1	0x1	0x1
Solaris 2.7	Not Implemented	0x1	0x1	0x1
Solaris 2.8	Not Implemented	0x1	0x1	0x1
HP-UX v10.20			Not Answering	
HP-UX v11.0	Not Answering	Not Answering	0x1	0x1
Compaq Tru64 v5.0	0x1	0x1	Not Answering	0x1
AIX 4.3	0x1	0x1	Not Answering	0x1
AIX 4.2.1	0x1	0x1	Not Answering	0x1
AIX 4.1	0x1	0x1	Not Answering	0x1
AIX 3.2	0x1	0x1	Not Answering	0x1
ULTRIX 4.2 – 4.5	0x0	0x0	0x0	0x0
OpenVMS v7.1-2	0x1	0x1	0x1	0x1
Windows 95	Not Answering	Not Answering		
Windows 98	Not Answering	0x0	0x0	0x1
Windows 98 SE	Not Answering	0x0	0x0	0x1
Windows ME	Not Answering	0x0	Not Answering	0x1
Windows NT 4 WRKS SP 3	Not Answering	Not Answering		
Windows NT 4 WRKS SP 6a	Not Answering	Not Answering	Not Answering	0x1
Windows NT 4 Server SP4	Not Answering	Not Answering	Not Answering	
Windows 2000 Professional	Not Answering	0x0	Not Answering	0x0
Windows 2000 Server	Not Answering	0x0	Not Answering	0x0

Appendix I: Using the Unused Bit

Operating System	Info. Request	Time Stamp Request	Address Mask Request	Echo Request
Debian GNU/ LINUX 2.2, Kernel 2.4 test 2	Not Answering	-	Not Answering	-
Redhat LINUX 6.2 Kernel 2.2.14	Not Answering	-	Not Answering	-
FreeBSD 4.0	Not Answering	-	Not Answering	-
FreeBSD 3.4	Not Answering	-	Not Answering	-
OpenBSD 2.7	Not Answering	-	Not Answering	-
OpenBSD 2.6	Not Answering	-	Not answering	-
NetBSD	Not Answering	-	Not Answering	-
BSDI BSD/OS 4.0	Not Answering	-	Not Answering	-
BSDI BSD/OS 3.1	Not Answering	-	Not Answering	-
Solaris 2.5.1	Not Answering	+	+	+
Solaris 2.6	Not Answering	+	+	+
Solaris 2.7	Not Answering	+	+	+
Solaris 2.8	Not Answering	+	+	+
HP-UX v10.20	-	-	Not Answering	-
HP-UX v11.0	Not Answering	Not Answering	+	+
Compaq Tru64 v5.0	-	-	Not Answering	-
Irix 6.5.3	Not Answering	-	Not Answering	-
Irix 6.5.8	Not Answering	-	Not Answering	-
AIX 4.1	-	-	Not Answering	-
AIX 3.2	-	-	Not Answering	-
ULTRIX 4.2 – 4.5	-	-	-	-
OpenVMS v7.1-2	-	-	-	-
Novell Netware 5.1 SP1	Not Answering	Not Answering	Not Answering	-
Novell Netware 5.0	Not Answering	Not Answering	Not Answering	-
Novell Netware 3.12	Not Answering	Not Answering	Not Answering	-
Windows 95	Not Answering	Not Answering		
Windows 98	Not Answering	-	-	-
Windows 98 SE	Not Answering	-	-	-
Windows ME	Not Answering		Not Answering	
Windows NT 4 WRKS SP 3	Not Answering	Not Answering		
Windows NT 4 WRKS SP 6a	Not Answering	Not Answering	Not Answering	-
Windows NT 4 Server SP4	Not Answering	Not Answering	Not Answering	-
Windows 2000 Professional	Not Answering	-	Not Answering	-
Windows 2000 Server	Not Answering	-	Not Answering	-

Appendix J: DF Bit Echoing

Operating System,	Info. Request	Time Stamp Request	Address Mask Request	Echo Request
Debian GNU/ LINUX 2.2. Kernel 2.4 test 2 (*)	Not Answering	+ (- DF)	Not Answering	+ (- DF)
Redhat LINUX 6.2 Kernel 2.2.14 (*)	Not Answering	+ (- DF)	Not Answering	+ (- DF)
FreeBSD 4.0 (*)	Not Answering	+ (+ DF)	Not Answering	+ (+ DF)
FreeBSD 3.4	Not Answering	+ (+ DF)	Not Answering	+ (+ DF)
OpenBSD 2.7	Not Answering	+ (+ DF)	Not Answering	+ (+ DF)
OpenBSD 2.6	Not Answering	+ (+ DF)	Not Answering	+ (+ DF)
NetBSD	Not Answering	+ (+ DF)	Not Answering	+ (+ DF)
BSDI BSD/OS 4.0	Not Answering	+ (+ DF)	Not Answering	+ (+ DF)
BSDI BSD/OS 3.1	Not Answering	+ (+ DF)	Not Answering	+ (+ DF)
Solaris 2.5.1	Not Answering			
Solaris 2.6	Not Answering	+ (+ DF)	+ (+ DF)	+ (+ DF)
Solaris 2.7 (*)	Not Answering	+ (+ DF)	+ (+ DF)	+ (+ DF)
Solaris 2.8	Not Answering	+ (+ DF)	+ (+ DF)	+ (+ DF)
HP-UX v10.20			Not Answering	
HP-UX v11.0	Not Answering	Not Answering	+ (+ DF)	+ (+ DF)
Compaq Tru64 v5.0 (*)		+ (+ DF)	Not Answering -	+ (+ DF)
Irix 6.5.3 (*)	Not Answering	+ (+ DF)	Not Answering	+ (+ DF)
Irix 6.5.8 (*)	Not Answering	+ (+ DF)	Not Answering	+ (+ DF)
AIX 4.1 (*)		+ (+ DF)	Not Answering	+ (+ DF)
AIX 3.2 (*)		+ (+ DF)	Not Answering	+ (+ DF)
ULTRIX 4.2 – 4.5 (*)		+ (- DF)	+ (- DF)	+ (- DF)
OpenVMS v7.1-2 (*)		+ (+ DF)	+ (+ DF)	+ (+ DF)
Novell Netware 5.1 SP1 (*)	Not Answering	Not Answering	Not Answering	+ (- DF)
Novell Netware 5.0 (*)	Not Answering	Not Answering	Not Answering	+ (- DF)
Novell Netware 3.12	Not Answering	Not Answering	Not Answering	+ (- DF)
Windows 95	Not Answering	Not Answering		
Windows 98 (*)	Not Answering	+ (- DF)	+ (- DF)	+ (+ DF)
Windows 98 SE (*)	Not Answering	+ (- DF)	+ (- DF)	+ (+ DF)
Windows ME (*)	Not Answering	+ (- DF)	Not Answering	+ (+ DF)
Windows NT 4 WRKS SP 3 (*)	Not Answering	Not Answering		
Windows NT 4 WRKS SP 6a (*)	Not Answering	Not Answering	Not Answering	+ (+ DF)
Windows NT 4 Server SP4	Not Answering	Not Answering	Not Answering	+ (+ DF)
Windows 2000 Professional (*)	Not Answering	+ (- DF)	Not Answering	+ (+ DF)
Windows 2000 Server (*)	Not Answering	+ (- DF)	Not Answering	+ (+ DF)

Appendix K: ICMP Error Message Echoing Integrity with ICMP Port Unreachable Error Message

Operating System	DF Bit set with the Reply?	IP Total Length	IP Identification	IP TTL field value	IP Header Checksum	UDP Checksum
LINUX Kernel 2.2.x	No	Same	Same	Changed according to hop count.	Changed because of new parameters.	Same
LINUX Kernel 2.4	No	Same	Same	Changed according to hop count.	Changed because of new parameters.	Same
FreeBSD 4.0	No	Same	Changed. The first two bits are flipped with the second pair. Gives a new value.	Changed according to hop count.	Changed because of new parameters.	Changed. Now equal to ZERO!
FreeBSD 4.11	No	Same	Same	Changed according to hop count.	Changed because of new parameters.	Changed. Now equal to ZERO!
BSDI 4.1	No	Changed (20 bytes more)	Same	Changed according to hop count	Changed. Now equals to ZERO!	Same
Sun Solaris 2.6	Yes	Same	Same	Changed according to hop count.	Changed because of new parameters.	Same
Sun Solaris 2.7	Yes	Same	Same	Changed according to hop count.	Changed because of new parameters.	Same
Sun Solaris 2.8[60]	Yes	Same	Same	Changed according to hop count.	Changed because of new parameters.	Same
HPUX 11.0	No -> Yes	Same	Same	Changed according to hop count.	Changed because of new parameters.	Same
Compaq Tru64	No	Same	Same	Changed according to hop count.	Changed because of new parameters.	Changed. Now equal to ZERO!
DG-UX 5.6	No	Same	Same	Changed according to hop	Changed because of new parameters.	Changed. Now equal to ZERO!

[60] The DF Bit is set.

				count.		
AIX 4.3 fp2, 4.3, 4.2.1	No	Changed (20 bytes more)	Same	Changed according to hop count	Changed because of new parameters.	Changed. Now equal to ZERO!
AIX 4.1	No	Changed (20 bytes more)	Same	Changed according to hop count	Changed because of new parameters.	Same
ULTRIX	No	Same	Changed. The first two bits are flipped with the second pair. Gives a new value.	Changed according to hop count	Changed. Now equals to ZERO!	Changed. Now equal to ZERO!
OpenVMS	No	Same	Changed. The first two bits are flipped with the second pair. Gives a new value.	Changed according to hop count	Changed. Now equals to ZERO!	Changed. Now equal to ZERO!
Microsoft windows 98 Mirosoft Windows 98SE	No	Same	Same	Changed. according to hop count.	Changed because of new parameters.	Same
Microsoft Windows ME	No	Same	Same	Changed according to hop count.	Changed because of new parameters.	Same
Microsoft Windows NT 4	No	Same	Same	Changed according to hop count.	Changed because of new parameters.	Same
Microsoft Windows 2000 Family	No	Same	Same	Changed according to hop count.	Changed because of new parameters.	Same

Appendix L: A Snort Rule Base for (more Advanced) Basic ICMP Traffic

The following generic ICMP basic Snort rule base is also available for download from: http://www.sys-security.com/archive/snort/icmp_rules/ICMP_basic_plus.

```
alert icmp any any -> any any (msg:"ICMP Echo Reply"; itype: 0; icode:
0;)
alert icmp any any -> any any (msg:"ICMP Echo Reply (Undefined Code!)";
itype: 0;)
alert icmp any any -> any any (msg:"ICMP Unassigned! (Type 1)"; itype:
1; icode: 0;)
alert icmp any any -> any any (msg:"ICMP Unassigned! (Tupe 1)
(Undefined Code)"; itype: 1;)
alert icmp any any -> any any (msg:"ICMP Unassigned! (Type 2)"; itype:
2; icode: 0;)
alert icmp any any -> any any (msg:"ICMP Unassigned! (Type 2)"
(Undefined Code); itype: 2;)
alert icmp any any -> any any (msg:"ICMP Destination Unreachable
(Network Unreachable)"; itype: 3; icode: 0;)
alert icmp any any -> any any (msg:"ICMP Destination Unreachable (Host
Unreachable)"; itype: 3; icode: 1;)
alert icmp any any -> any any (msg:"ICMP Destination Unreachable
(Protocol Unreachable)"; itype: 3; icode: 2;)
alert icmp any any -> any any (msg:"ICMP Destination Unreachable (Port
Unreachable)"; itype: 3; icode: 3;)
alert icmp any any -> any any (msg:"ICMP Destination Unreachable
(Fragmentation Needed and DF bit was set)"; itype: 3; icode: 4;)
alert icmp any any -> any any (msg:"ICMP Destination Unreachable
(Source Route Failed)"; itype: 3; icode: 5;)
alert icmp any any -> any any (msg:"ICMP Destination Unreachable
(Destination Network Unknown)"; itype: 3; icode: 6;)
alert icmp any any -> any any (msg:"ICMP Destination Unreachable
(Destination Host Unknown)"; itype: 3; icode: 7;)
alert icmp any any -> any any (msg:"ICMP Destination Unreachable
(Source Host Isolated)"; itype: 3; icode: 8;)
alert icmp any any -> any any (msg:"ICMP Destination Unreachable
(Communication with Destination Network is Administratively
Prohibited)"; itype: 3; icode: 9;)
alert icmp any any -> any any (msg:"ICMP Destination Unreachable
(Communication with Destination Host is Administratively Prohibited)";
itype: 3; icode: 10;)
alert icmp any any -> any any (msg:"ICMP Destination Unreachable
(Network Unreachable for Type of Service)"; itype: 3; icode: 11;)
alert icmp any any -> any any (msg:"ICMP Destination Unreachable (Host
Unreachable for Type of Service)"; itype: 3; icode: 12;)
alert icmp any any -> any any (msg:"ICMP Destination Unreachable
(Communication Administratively Prohibited)"; itype: 3; icode: 13;)
alert icmp any any -> any any (msg:"ICMP Destination Unreachable (Host
Precedence Violation)"; itype: 3; icode: 14;)
alert icmp any any -> any any (msg:"ICMP Destination Unreachable
(Precedence Cutoff in effect)"; itype: 3; icode: 15;)
alert icmp any any -> any any (msg:"ICMP Destination Unreachable
(Undefined Code!)"; itype: 3;)
alert icmp any any -> any any (msg:"ICMP Source Quench"; itype: 4;
icode: 0;)
```

```
alert icmp any any -> any any (msg:"ICMP Source Quench (Undefined
Code!)"; itype: 4;)
alert icmp any any -> any any (msg:"ICMP Redirect (for Network or
Subnet)"; itype: 5; icode: 0;)
alert icmp any any -> any any (msg:"ICMP Redirect (for Host)"; itype:
5; icode: 1;)
alert icmp any any -> any any (msg:"ICMP Redirect (for TOS and
Network)"; itype: 5; icode: 2;)
alert icmp any any -> any any (msg:"ICMP Redirect (for TOS and Host)";
itype: 5; icode: 3;)
alert icmp any any -> any any (msg:"ICMP Redirect (Undefined Code!)";
itype: 5;)
alert icmp any any -> any any (msg:"ICMP Alternate Host Address";
itype: 6; icode: 0;)
alert icmp any any -> any any (msg:"ICMP Alternate Host Address
(Undefined Code!)"; itype: 6;)
alert icmp any any -> any any (msg:"ICMP Unassigned! (Type 7)"; itype:
7; icode: 0;)
alert icmp any any -> any any (msg:"ICMP Unassigned! (Type 7)
(Undefined Code!)"; itype: 7;)
alert icmp any any -> any any (msg:"ICMP Echo Request"; itype: 8;
icode: 0;)
alert icmp any any -> any any (msg:"ICMP Echo Request (Undefined
Code!)"; itype: 8;)
alert icmp any any -> any any (msg:"ICMP Router Advertisment"; itype:
9; icode: 0;)
alert icmp any any -> any any (msg:"ICMP Router Advertisment (Undefined
Code!)"; itype:9 ;)
alert icmp any any -> any any (msg:"ICMP Router Selection"; itype: 10;
icode: 0;)
alert icmp any any -> any any (msg:"ICMP Router Selection (Undefined
Code!)"; itype: 10;)
alert icmp any any -> any any (msg:"ICMP Time-To-Live Exceeded in
Transit"; itype: 11; icode: 0;)
alert icmp any any -> any any (msg:"ICMP Fragment Reassembly Time
Exceeded"; itype: 11; icode: 1;)
alert icmp any any -> any any (msg:"ICMP Time Exceeded (Undefined
Code!)"; itype: 11;)
alert icmp any any -> any any (msg:"ICMP Parameter Problem Code 0
(unspecified Error)"; itype: 12; icode: 0;)
alert icmp any any -> any any (msg:"ICMP Parameter Problem Code 1
(Missing a Requiered Option)"; itype: 12; icode: 1;)
alert icmp any any -> any any (msg:"ICMP Parameter Problem Code 2 (Bad
Length)"; itype: 12; icode: 2;)
alert icmp any any -> any any (msg:"ICMP Parameter Problem (Undefined
Code!)"; itype: 12;)
alert icmp any any -> any any (msg:"ICMP Timestamp Request"; itype: 13;
icode: 0;)
alert icmp any any -> any any (msg:"ICMP Timestamp Request (Undefined
Code!)"; itype: 13;)
alert icmp any any -> any any (msg:"ICMP Timestamp Reply"; itype: 14;
icode: 0;)
alert icmp any any -> any any (msg:"ICMP Timestamp Reply (Undefined
Code!)"; itype: 14;)
alert icmp any any -> any any (msg:"ICMP Information Request"; itype:
15; icode: 0;)
```

```
alert icmp any any -> any any (msg:"ICMP Information Request (Undefined
Code!)"; itype: 15;)
alert icmp any any -> any any (msg:"ICMP Information Reply"; itype: 16;
icode: 0;)
alert icmp any any -> any any (msg:"ICMP Information Reply (Undefined
Code!)"; itype: 16;)
alert icmp any any -> any any (msg:"ICMP Address Mask Request"; itype:
17; icode: 0;)
alert icmp any any -> any any (msg:"ICMP Address Mask Request
(Undefined Code!)"; itype: 17;)
alert icmp any any -> any any (msg:"ICMP Address Mask Reply"; itype:
18; icode: 0;)
alert icmp any any -> any any (msg:"ICMP Address Mask Reply (Undefined
Code!)"; itype: 18;)
alert icmp any any -> any any (msg:"ICMP Reserved for Security (Type
19)"; itype: 19; icode: 0;)
alert icmp any any -> any any (msg:"ICMP Reserved for Security (Type
19) (Undefined Code!)"; itype: 19;)
alert icmp any any -> any any (msg:"ICMP Traceroute"; itype: 30; icode:
0;)
alert icmp any any -> any any (msg:"ICMP Traceroute (Undefined Code!";
itype: 30;)
alert icmp any any -> any any (msg:"ICMP Datagram Conversion Error";
itype: 31; icode: 0;)
alert icmp any any -> any any (msg:"ICMP Datagram Conversion Error
(Undefined Code!)"; itype: 31;)
alert icmp any any -> any any (msg:"ICMP Mobile Host Redirect"; itype:
32; icode: 0;)
alert icmp any any -> any any (msg:"ICMP Mobile Host Redirect
(Undefined Code!)"; itype: 32;)
alert icmp any any -> any any (msg:"ICMP IPV6 Where-Are-You"; itype:
33; icode: 0;)
alert icmp any any -> any any (msg:"ICMP IPV6 Where-Are-You (Undefined
Code!)"; itype: 33;)
alert icmp any any -> any any (msg:"ICMP IPV6 I-Am-Here"; itype: 34;
icode: 0;)
alert icmp any any -> any any (msg:"ICMP IPV6 I-Am-Here (Undefined
Code!"; itype: 34;)
alert icmp any any -> any any (msg:"ICMP Mobile Registration Request";
itype: 35; icode: 0;)
alert icmp any any -> any any (msg:"ICMP Mobile Registration Request
(Undefined Code!"; itype: 35;)
alert icmp any any -> any any (msg:"ICMP Mobile Registration Reply";
itype: 36; icode: 0;)
alert icmp any any -> any any (msg:"ICMP Mobile Registration Reply
(Undefined Code!)"; itype: 36;)
alert icmp any any -> any any (msg:"ICMP SKIP"; itype: 39; icode: 0;)
alert icmp any any -> any any (msg:"ICMP SKIP (Undefined Code!"; itype:
39;)
alert icmp any any -> any any (msg:"ICMP Photuris Code 0 (Reserved)";
itype: 40; icode: 0;)
alert icmp any any -> any any (msg:"ICMP Photuris Code 1 (Unknown
Security Parameters Index)"; itype: 40; icode: 1;)
alert icmp any any -> any any (msg:"ICMP Photuris Code 2 (Valid
Security Parameters, But Authentication Failed)"; itype: 40; icode: 2;)
alert icmp any any -> any any (msg:"ICMP Photuris Code 3 (Valid
Security Parameters, But Decryption Failed)"; itype: 40; icode: 3;)
```

```
alert icmp any any -> any any (msg:"ICMP Photuris (Undefined Code!)";
itype: 40;)
alert icmp any any -> any any (msg:"ICMP Unknown Type";)
```

For corrections/additions/suggestions for this research paper, please send email to ofir@sys-security.com. Further Information and updates would be posted to http://www.sys-security.com.

Thank you for reading

Ofir Arkin

Founder

The Sys-Security Group

Sys-Security Group
BECAUSE SECURITY IS NOT TRIVIAL

http://www.sys-security.com
ofir@sys-security.com

EXHIBIT 'G'

Welcome to Sys-Security.com

Home

Sys-Security Group

BECAUSE SECURITY IS NOT TRIVIAL

Sys-Security.com is a web site dedicated to computer security research. It is the home of the "ICMP Usage In Scanning" research project.

Navigation
- Advisories
- Articles Mentioned
- Conferences
- News
- Pictures
- Published Papers
- Upcoming Events

- Books
- Links
- Tools

Projects
- The ICMP Project

- VoIP
- Xprobe

Latest News

September 19, 2002
"The Trivial Cisco IP Phones Compromise" paper was released...

August 20, 2002
Advisory: More Vulnerabilities with Pingtel xpressa SIP-based IP Phones...

August 9, 2002
Xprobe2 version 0.1 release candidate 1 has been released...

The Trivial Cisco IP Phones Compromise

The following paper lists several severe vulnerabilities with Cisco systems' SIP-based IP Phone 7960 and its supporting environment. These vulnerabilities lead to complete control of a user's credentials, the total subversion of a user's settings for the IP Telephony network, and the ability to subvert the entire IP Telephony environment. Malicious access to a user's credentials could enable "Call Hijacking", "Registration Hijacking", "Call Tracking", and other voice related attacks. The vulnerabilities exist with any deployment scenario, but this paper deals specifically with large scale deployments as recommended by Cisco.

Full Details in PDF format ~200kb
Full Details in PDF Zipped format

More Vulnerabilities with Pingtel xpressa SIP-based IP Phones

Pingtel (http://www.pingtel.com) develops intelligent Java-based voice-over-IP phones and softphones for service providers and enterprises.

Using the vulnerabilities enumerated within this advisory it is possible to jeopardize critical telephony infrastructure based on Pingtel's xpressa SIP-based IP phones and softphones. Additionally, certain vulnerabilities allow an attacker to take complete control over an IP Phone or a softphone node either directly or by circumventing other SIP entities on the network by abusing the 'node's credentials'.

The most severe issue discussed is the way an attacker can exploit vulnerabilities with MyPingtel portal (http://my.pingtel.com) to subvort a VoIP infrastructure which includes IP Phones and/or softphones from Pingtel... >>

Full Details in PDF format ~500kb
Full Details in HTML format
Moderated version in txt format

Xprobe2

Xprobe2 is an active operating system fingerprinting tool with a different approach to operating system fingerprinting. Xprobe2 rely on fuzzy signature matching, probabilistic guesses, multiple matches simultaneously, and a signature database.

The tool's release is accompanied by a white paper explaining our fuzzy approach to operating system fingerprinting and the various problems facing other remote active operating system fingerprinting tools available today, which we have tried to remedy.

The current development code (xprobe2 0.1 rc1) is available from:
http://www.sys-security.com/archive/tools/xprobe2/xprobe2-0.1rc1.tar.gz

The White Paper is available from:
http://www.sys-security.com/archive/papers/Xprobe2.pdf

ICMP Usage In Scanning

The ICMP protocol may seem harmless at first glance, but in terms of security ICMP is one of the most controversial protocols with the TCP/IP suite. The risks involved in implementing the ICMP protocol in a network regarding scanning are the subject of this research. Issues discussed range from understanding the different ICMP messages, reconnaissance (host detection, inverse mapping, Active and passive operating system fingerprinting) and more. >>

Xprobe2

Navigate Sys-Security.com:
Choose one: ▾
Go

For corrections/additions/suggestions for this page, please send email to: ofir@sys-security.com

Trace-Back

Trace-Back is *"A Concept for Trac[illegible]nd Profiling Malicious Computer Attackers"*
From the Introduction:

> "In the computer security arena, every now and then, a vulnerability comes along causing a significant impact. The impact of a vulnerability is based on factors such as popularity of the vulnerable platform and the ease of exploitation of the vulnerability. Lots of research gets done on a vulnerability, beginning from its origin to the various permutations and combinations of exploit code that come out subsequently. In recent years, we have seen self-propagating exploit code (in other words, worms) becoming quite popular.
>
> Very little is known about the events taking place in the time period between the instance that a vulnerability gets discovered by an individual or a small group of individuals, and the moment when exploit code becomes publicly available on the Internet. To zero in on the origins of a particular piece of exploit code is quite a daunting task. Very little research has been done on the subject outside of government or military organizations. Tracing back origins is a very tricky task, especially if one has to reconstruct events backwards. This paper addresses this very issue - trying to roll the film reel backwards from the time the exploit code becomes widespread in public, and filling in the blank frames to the beginning of the movie. This may not be the ultimate "big-bang" theory of the exploit universe, but it provides us with new viewpoints on exploits and their originators..."